Universidad
Carlos III de Madrid

Departamento de Ingeniería Telemática

Aplicación Android para obtener información de tráfico en carreteras

Trabajo Fin de Grado

Autor:
Álvaro González Caballero

Tutor:
Mª Celeste Campo Vázquez

Grado en Ingeniería Telemática
Escuela Politécnica Superior Universidad Carlos III de Madrid
Leganés, Septiembre de 2016

Título: Aplicación Android para obtener información de tráfico en carreteras.

Autor: D. Álvaro González Caballero.

Tutora: Dña. Mª Celeste Campo Vázquez.

EL TRIBUNAL

Presidente: D. David Larrabeiti López

Secretario: D. Rodolfo Cuerno Rejado

Vocal: D. Yoel Gustavo Yera Mora

Suplente: D. Pedro José Muñoz Merino

Realizado el acto de defensa y lectura del Trabajo de Fin de Grado el 7 de octubre de 2016, en Leganés, Escuela Politécnica Superior de la Universidad Carlos III de Madrid.

Agradecimientos

Este documento simboliza el fin de una etapa muy importante en mi vida, tanto en lo académico como en lo personal, por ello no quiero dejar pasar la ocasión para dar las gracias:

A mis padres, Coro y Jesús, por vuestro esfuerzo para darme una educación, pero sobre todo por vuestro cariño y comprensión.

A mi tate, Jesús, por ser siempre el hombro donde apoyarme, por tus consejos y cariño.

A Elsa, por aparecer, por enseñarme que puedo con todo y ayudarme siempre. Sin ti todo esto no hubiera sido posible.

A mis abuelos, estén o no físicamente siempre os llevo en el corazón. En especial a mi abuela Paca, por estar siempre pendiente de mis estudios y de recordarme que si vagueo viene "el tío de las rebajas".

A mis amigos, porque aunque pasen los años, seguís estando a mi lado. ¡Y los que nos quedan!

Gracias

Resumen

Desde inicios del siglo XX hasta la actualidad, se ha hecho patente el incremento de la población de núcleos urbanos en España. Este fenómeno se conoce con el nombre de éxodo rural y se fundamenta en la búsqueda por parte de la población, de lugares con mayor nivel de industrialización (lo cual supone oportunidades de trabajo). Las grandes ciudades han tenido que afrontar diferentes retos. Uno de ellos es el gran número de vehículos que circulan por sus calles y carreteras, llegando a provocar en hora punta numerosos y kilométricos atascos.

Por otro lado, el desarrollo tecnológico que se vive actualmente, ha propiciado la aparición de dispositivos móviles, con grandes capacidades de cómputo y numerosos sensores. Vivimos en la era de las comunicaciones a través de la red.

El presente Trabajo de Fin de Grado busca dar una solución a este problema del tráfico. Para ello se aprovecha el fenómeno de los dispositivos móviles, que por su tamaño y su versatilidad, están presentes en el día a día de millones de personas.

Son numerosas las herramientas que proporcionan los dispositivos móviles: sensores de aceleración, de localización, de movimiento, de sensibilidad a la luz. Esta solución para la detección de tráfico, en especial de eventos de congestión, se basa en la utilización de diferentes sensores. Por esta razón se diferencia de otras soluciones del mercado, al detectar los eventos de forma local en el propio dispositivo. Se trata de demostrar que existen soluciones alternativas, que no sólo ofrecen la misma fiabilidad, sino que también proporcionan mayor privacidad, además de una utilización de los recursos de forma sensata.

Abstract

Since the beginning of 20th century, the higher level of population in urban areas has become a fact in Spain. We know this phenomenon by the name of rural depopulation. This phenomenon is based on people's search of a better way to improve their life (they are looking forward to a better job, in places with a high level of industrialisation). This supposes a goal for big cities. They face a number of challenges due to the increase of population. One of these challenges is the big number of vehicles travelling around the different streets or roads. This is the main cause of traffic congestion during rush hours.

On the other hand, we live in a technological epoch where thanks to the big and rapid development of new technologies, there are available so many different types of mobile gadgets. These gadgets have a big computational capacity and a lot of sensors. We live in the "always on" age, and we use telecommunications for everything

This Bachelor Thesis tries to look for a solution to the traffic problem. For that purpose, it uses the mobile devices phenomenon. The number of devices, the size, and the adaptability of this product are the main reasons why they are on millions of people life.

Mobile devices provide a large number of different tools, as for example an accelerometer sensor, a location sensor, a movement sensor, or a light sensitivity sensor. This solution, for traffic congestion events, is based on the use of different sensors. The main difference with other solutions is the use of these sensors to detect events in a local way. The idea is to show, that is possible to create alternative solutions, that are more safety for the users, with the same accuracy.

Lista de acrónimos:

AEMET - Agencia Estatal de Meteorología.

API - Application Programming Interface.

APP - Aplicación móvil.

AVD - Android Virtual Device.

BAM - Banda Ancha Móvil.

CU - Caso de uso.

DGT - Dirección General de Tráfico.

UE - Unión Europea.

ERS - Especificación Requisitos de Software.

FM - Frecuencia modulada.

GPS - Sistema de posicionamiento global.

HTTP - Hypertext Transfer Protocol.

IDE - Integrated Development Environment.

PMUS - Plan de Movilidad Urbana Sostenible de la ciudad de Madrid.

PS - Prueba de Software.

RDS - Radio Data System.

RS - Requisito de Software.

SO - Sistema Operativo.

TFG - Trabajo de Fin de Grado.

XML - Extensible Markup Language.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Nota al lector de este TFG:

Antes de comenzar con la lectura del proyecto, es necesario poner al lector en antecedentes. El presente TFG surge de la inquietud por encontrar soluciones cotidianas, a problemas cotidianos, aprovechando el auge de los dispositivos móviles, y su gran adopción en nuestro país. Hoy en día, se cuentan por millones los conductores que llevan un Smartphone en su bolsillo, y esa es la ventaja que pretende aprovechar este proyecto.

De forma paralela a este proyecto, mi compañera Elsa Gómez Martínez ha desarrollado su TFG con título: "*Aplicación Android para obtener información de las carreteras*" [43]. Ambos, hacemos uso de los diferentes sensores que proporcionan los Smartphone, para desarrollar nuestras soluciones. Como eje común, además del dispositivo acelerómetro correctamente calibrado, ambos trabajos se basan en la filosofía de desarrollar soluciones de bajo coste, y que repercutan en una mejora de la privacidad del usuario, y de los recursos necesarios, todo ello manteniendo la calidad que ofrecen las soluciones actuales del mercado.

Debido al desconocimiento en el lenguaje de programación Android (al inicio de este trabajo), y a la necesidad de utilizar un acelerómetro calibrado, se decidió desarrollar distintos prototipos de forma común, que nos han permitido a ambos desarrollar correctamente nuestras soluciones finales, y facilitar el aprendizaje de la programación en Android. De forma concreta, se han desarrollado en común los siguientes módulos:

- Introducción y aprendizaje del sistema operativo Android.
- Aplicación para aprender el uso del GPS.
- Aplicación para aprender el uso del acelerómetro.
- Aplicación que calibra el acelerómetro, reorientándolo.
- Aplicación para el guardado de datos en ficheros.

Adicionalmente, para facilitar la tarea de identificación de los módulos realizados de forma común, con las secciones a las que estos corresponden dentro de la memoria, se realizará una anotación al pie de página en aquellas secciones que lo necesiten.

Por tanto, ambos trabajos cuentan con una base y una estructura común, que quedará reflejada en sendas memorias, pero dan solución final a dos problemas bien diferenciados, utilizando diferentes técnicas como leeremos a continuación.

English version available for this chapter.

1. Introducción y objetivos

En este primer capítulo se realizará una introducción sobre el tema a tratar: el tráfico, y la obtención de información a través de las nuevas tecnologías. Para ello se explicarán los objetivos perseguidos, las fases de desarrollo del proyecto, los recursos empleados y por último la estructura del documento a modo de guía del mismo.

1.1. Introducción

El desplazamiento de personas o mercancías es un hecho ligado a la existencia del hombre. Desde su origen, el tránsito vehicular (o tráfico) es un fenómeno histórico, social, económico y jurídico que afecta a la mayor parte de las sociedades. Su causa es el flujo de vehículos que circulan por una calle, autopista o vía en general. [1]

Se puede realizar un estudio de diversos fenómenos que están relacionados con el tráfico, por ejemplo: accidentes de tráfico, uso de semáforos, cruces en avenidas, etc. Aunque el caso que nos ocupa y que estudiaremos a lo largo de este Trabajo de Fin de Grado (TFG), es el de la congestión vehicular. [2]

Figura 1.1: Congestión vehicular en la salida de Madrid por la autovía A4.

La congestión vehicular [3] (comúnmente llamada atasco o tráfico), ocurre debido a la saturación de las vías. Si la demanda de vehículos para una vía es superior a la tasa que dicha vía puede asimilar, se produce el llamado tráfico o embotellamiento. Este fenómeno ocurre en horas punta, causando pérdida de tiempo y frustración en los conductores, además de las

correspondientes pérdidas económicas, por consumo de combustible y disminución de la productividad, e incrementando la contaminación.

Ranking	Áreas metropolitanas	Total horas perdidas 2014
1	Barcelona	25
2	Madrid	22
3	Sevilla	18
4	Bilbao	16
5	Zaragoza	12
6	Valencia	11

Tabla 1-1: Ranking por horas perdidas en las áreas metropolitanas más congestionadas de España, durante 2014. (Fuente INRIX)

Debido al carácter global del problema del tráfico, y el afán del hombre por evolucionar, a lo largo de la historia se han dado soluciones con diferentes enfoques. Algunas van ligadas al desarrollo tecnológico sufrido en las últimas décadas, donde la miniaturización de los componentes ha permitido la creación de dispositivos electrónicos fáciles de transportar a cualquier sitio. Uno de los mejores ejemplos es el de los dispositivos móviles, por su gran capacidad de procesamiento, así como por contar con diversos sensores (acelerómetro, GPS, conectividad de distintos tipos, etc.).

Estas características, unidas a su facilidad en el manejo, han convertido a los dispositivos móviles en uno de los objetos más útiles para los conductores. El uso de sus sensores, o sus capacidades para la conectividad, hacen posible la comunicación entre usuarios, pudiendo así compartir todo tipo de información.

A continuación, resumiré algunas soluciones al problema del tráfico, y en concreto a la detección de eventos de congestión. Son soluciones diversas basadas en diferentes enfoques, donde algunas de ellas utilizan como eje principal dispositivos electrónicos.

1.1.1. Soluciones tecnológicas

En la actualidad, existen varias soluciones tecnológicas, cuyo fin es la detección y prevención de atascos a sus usuarios. Algunos ejemplos de este tipo de tecnologías son:

Aplicaciones para Smartphone: Permiten al usuario compartir información del tráfico en tiempo real. Existen tres tipos de aplicaciones: colaborativas, basadas en algoritmos y basadas en la toma masiva de datos. Se detallarán estas soluciones en el capítulo 2.

Tecnología autónoma: Un vehículo autónomo es capaz de percibir el medio que le rodea, actuando así en consecuencia. Conectado en tiempo real a la red, puede recibir información del tráfico que le permitirá elegir la ruta más óptima, o la velocidad adecuada a cada momento.

Radio Data System (RDS): Se trata de una tecnología que permite añadir información a los programas de radio en FM (frecuencia modulada). Una de sus aplicaciones es la recepción de información relacionada con el tráfico. Permite, de forma automática, el cambio a una emisora que emita información de tráfico, mientras se tenga el dispositivo de radio encendido.

1.1.2. Soluciones urbanísticas

La congestión vehicular se manifiesta con frecuencia en los accesos a las grandes urbes. Algunos hechos objetivos que fundamentan dicho problema son:

- El 72% de los ciudadanos europeos residen en zonas urbanas.
- En las ciudades se consume el 75% de la energía y es donde se genera el 85% de la riqueza de la Unión Europea (UE).
- El 98% de los desplazamientos corresponden a desplazamientos urbanos inferiores a 50 kilómetros.

Para atajar este problema, ciudades europeas como Barcelona, Londres o Madrid elaboran distintas soluciones. Un buen ejemplo de ello es el plan de movilidad Urbana Sostenible de la ciudad de Madrid (PMUS Madrid) [4].

Un PMUS, se define en la guía europea *Developing and implementing a Sustainable urban mobility plan,* como un plan estratégico, diseñado para satisfacer las necesidades de movilidad de personas y empresas en las ciudades y sus alrededores, buscando una mejora en la calidad de vida.

En su PMUS de diciembre de 2014, que se presenta para concretar planteamientos desarrollados desde 2006, Madrid realiza un diagnóstico de la situación actual para posteriormente definir objetivos y medidas, que permitan gestionar la movilidad urbana de forma sostenible.

Figura 1.2: Objetivos PMUS Madrid 2014. (Fuente PMUS)

Para conseguir sus objetivos de seguridad vial (reducción de accidentes y reducción de víctimas por accidente), sostenibilidad (mejora en la calidad del aire, la eficiencia energética, etc.) y universalidad (transporte público universal, movilidad peatonal y ciclista accesible y universal, y espacio público de conveniencia) se definen las siguientes medidas:

- Fomento de la movilidad peatonal.
- Fomento del transporte público colectivo.
- Fomento de la movilidad ciclista.
- Mejora de la movilidad en moto.
- Optimización del servicio del taxi.
- Incorporación al sistema de transportes de nuevos modelos de movilidad colaborativa.
- Mejora de la accesibilidad a todos los modos de transporte
- Mejora de las condiciones de intermodalidad.
- Mejora de la gestión de la demanda de vehículo privado.
- Mejoras en la gestión de la circulación.
- Optimización de la distribución urbana de mercancías.
- Promoción de energías limpias en la tecnología de los vehículos.
- Mejoras de la gestión del transporte turístico y discrecional.
- Comunicar y formar para el cambio de hábitos.
- Impulsar la implicación del sector privado en la planificación y gestión de la movilidad.

1.1.3. Otras soluciones

Existen, además, otras soluciones que ayudan a paliar el problema de la congestión vehicular, algunas de forma indirecta. Entre este tipo de soluciones encontramos:

Dirección General de Tráfico (DGT): Organismo Jefatura Central de Tráfico en España, surge en 1959 para hacer frente a la creciente motorización del país. En la actualidad, es el encargado de velar por la seguridad y ordenar el tráfico. Consigue estas tareas mediante la realización de campañas de sensibilización, de concienciación, ofreciendo previsiones, recomendaciones y el establecimiento de restricciones de circulación, etc.

Uso compartido: Además del transporte público colectivo, existen iniciativas para el uso compartido de vehículos privados, que no sólo benefician el bolsillo del consumidor, sino también beneficia, en gran medida, a disminuir los problemas de tráfico.

1.2. Objetivos

El TFG desarrollado pretende obtener información sobre el estado de las carreteras mediante una aplicación en Android. La mayor parte de aplicaciones existentes en el mercado, basan su funcionamiento en el envío de datos obtenidos a un servidor, donde se procesan y analizan. Este trabajo, busca dar el enfoque contrario. Se trata de recoger los datos en el propio dispositivo, aprovechando los sensores de los que dispone, para después utilizar su gran capacidad de procesamiento y detectar los posibles eventos de congestión.

Teniendo en cuenta todos estos requisitos, se han definido los siguientes objetivos.

1.2.1. Objetivos del proyecto

El objetivo principal es el desarrollo de una aplicación Android basada en el análisis de datos recogidos mediante sensores, que permita la detección del tráfico de forma segura, y que respete la privacidad del usuario en la mayor medida posible.

Si partimos del funcionamiento de las soluciones más extendidas, observamos que el envío continuo de datos a servidores provoca dos puntos débiles: La privacidad del usuario queda en entredicho, y el consumo del dispositivo puede ser mayor del deseado (tanto para la batería, como para los datos). Buscando solucionar ambos problemas definimos, a continuación, los siguientes objetivos secundarios:

- Se desarrollará la aplicación teniendo en cuenta la protección de la privacidad del usuario. Se pretende detectar únicamente los puntos donde existe congestión en el tráfico, sin necesidad de conocer la ruta completa que sigue el usuario. De esta forma establecemos que la detección de tráfico se efectuará de forma local en el dispositivo, sin necesidad de envío de datos a un servidor.
- La gestión de recursos deberá ser eficaz, lo que permita un consumo de batería y datos lo más bajo posible.
- Se destaca la solución implementada por encima de la aplicación en sí. Se desarrolla la solución, pensando en su posible implementación en sistemas de información de tráfico.

1.2.2. Objetivos personales

La elección de desarrollar software en Android se fundamenta en la persecución de los siguientes objetivos personales:

- Aprender a desarrollar e implementar aplicaciones móviles, en un sistema como Android con las ventajas que ello supone (serán vistas en capítulos siguientes).
- Estudiar y utilizar metodologías ágiles de desarrollo como Scrum [29], fundamentales hoy en día para la realización de proyectos disminuyendo riesgos.

1.3. Fases de desarrollo

El desarrollo del proyecto se ha llevado a cabo en diferentes fases, todas ellas se describen, a continuación, siguiendo un orden cronológico ascendente.

Fase 1 – Análisis del problema: Durante esta primera fase se analiza el problema en cuestión y se establecen soluciones alternativas a las ya existentes. Para ello se tendrán en cuenta nuestros objetivos. Se trata de una fase de documentación del problema, que nos permitirá abordar con mayor éxito las siguientes fases.

Algunos de los documentos consultados durante esta fase, son los artículos nombrados a continuación:

- P. Mohan, V.N. Padmanabhan and R. Ramjee. “TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones”, *Technical Report MSR-TR-2008-59*, Bangalore, India, 2008. [27]
- S. Ayub, A. Bahraminisaab y B. Honary. “A sensor Fusion Method for Smartphone Orientation Stimation”, presentada en 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, Junio 2012. [31]

- R. Bhoraskar, “Traffic and Road Condition Estimation using Smartphone Sensors.”, Bachelor Thesis, Department of Computer Science and Engineering, Indian Institue of Technology, Bombay, Mumbai, India, 2011. [30]

Fase 2 – Formación: Esta fase está dedicada a la formación en las herramientas necesarias para la realización del proyecto. Android Studio es el Integrated Development Environment (IDE) oficial para Android. Para el desarrollo en esta plataforma es necesario estar familiarizado con los lenguajes XML y Java, además de con la biblioteca de Android Studio.
Los recursos utilizados para la formación en Android han sido:

- C. Campo Vázquez y C. García Rubio, Curso de Android de la asignatura Aplicaciones móviles, Universidad Carlos III de Madrid, Leganés, 2015. [32]
- “Web para desarrolladores de Android”, *Google*. [17]
- S. Gómez Oliver, *Curso Programación Android*, 3th ed. Publicación libre, 2015. [33]

Fase 3 – Desarrollo: Con los conocimientos adquiridos en la Fase 2, durante esta nueva fase, se procede al inicio del desarrollo. Para ello se comienza con la creación de programas utilizando los sensores necesarios. Se trata de poner en práctica lo aprendido, generando distintas y pequeñas aplicaciones que utilicen los recursos necesarios para nuestra aplicación final.

Es en esta fase donde aplicar la metodología Scrum nos será de gran ayuda. Crearemos un desarrollo modular, en múltiples iteraciones, cumpliendo las diferentes fases de Scrum en cada una de ellas. Esto nos permitirá desarrollar la aplicación mientras aprendemos el funcionamiento del sistema operativo (SO) y su lenguaje. Además, podremos probar y analizar datos preliminares que nos ayudarán en el desarrollo de la aplicación final.

Fase 4 – Aplicación final: Durante esta fase se procede al desarrollo de una aplicación final que cumpla los objetivos establecidos. Se trata de la integración de aquellos pequeños módulos desarrollados durante la Fase 3.

Fase 5 – Pruebas y Análisis de datos: Esta fase servirá para probar la aplicación final. Se medirán los valores de los sensores utilizados para establecer umbrales. Además, se realizarán pruebas en un entorno real para posteriormente analizar los datos obtenidos y verificar el correcto comportamiento de la aplicación.

Fase 6 – Documentación: Utilizando documentación preliminar, en esta fase se procederá a la creación de la documentación final del proyecto. Para ello se redactará la versión final de la presente memoria, así como un manual de usuario y la planificación y gestión del proyecto.

1.4. Recursos empleados

A continuación, se enumeran los distintos recursos empleados para la consecución de este proyecto. Se realiza una distinción entre hardware, software y otros medios.

Hardware:

- Smartphone Android, marca BQ y modelo Aquaris A4.5.
- Smartphone Android, marca Samsung y modelo A5.
  Ambos dispositivos se han utilizado para probar en un entorno real la aplicación. Elegidos por su precio y sus características.
- Ultrabook marca Apple y modelo Macbook Pro retina 13'. Se ha utilizado para el desarrollo de la aplicación y análisis de los datos.

Software:

- Mac OS X 'El Capitán' 10.11.4. Es el SO por defecto para el Ultrabook elegido.
- Android Marshmallow 6.0. Es el SO por defecto con los Smartphone elegidos.
- Android Studio (IDE). Es el entorno de desarrollo proporcionado por Google.
- Paquete ofimática Microsoft Office 2011. En concreto las herramientas Word y Excel, utilizadas durante la redacción de esta memoria.
- Biblioteca Here WeGo. En concreto su API para tráfico, utilizada para la obtención de la velocidad máxima en carreteras.
- Matlab R2015b. Elegida por su gran capacidad de manejo de datos, se utiliza en el análisis de los datos recogidos por la aplicación.

Otros:

- Vehículo: Peugeot 406.
- Vehículo: Opel Vectra.
- Vehículo: Audi Q5.
- Vehículo: Citroën C8.
  Se utilizan estos vehículos para la circulación en carreteras, probando la aplicación en un entorno real. Su elección se basa en contar con distintos segmentos del mercado, que pueden responder de forma distinta a las aceleraciones medidas con el acelerómetro (por su potencia, tipo de conducción, etc.).
- Soporte para la sujeción de dispositivos móviles.

1.5. Estructura del documento

En este apartado se realiza una descripción general de la estructura del documento.

1) **Introducción y objetivos:** En este primer capítulo se realizará una introducción sobre el tema a tratar: el tráfico, y la obtención de información a través de las nuevas tecnologías, en concreto para eventos de congestión en la red viaria. Para ello se tratarán los objetivos perseguidos, las fases de desarrollo del proyecto, los recursos empleados y por último la estructura del documento a modo de guía del mismo.

2) **Estado del arte:** En este capítulo se realizará una revisión de las bases técnicas del TFG. Para ello se analizarán las soluciones que existen actualmente en el mercado al problema de la congestión viaria, los dispositivos y tecnologías disponibles, así como el desarrollo de aplicaciones móviles. Todo ello sirve como investigación que fundamenta las decisiones tomadas durante la realización del trabajo.

3) **Análisis y diseño de la solución técnica:** En este capítulo se analiza y se describe el diseño de la solución técnica. Se realiza a través de la descripción de la arquitectura para, a continuación, presentar los requisitos a resolver y los casos de uso a tratar.

4) **Implementación del sistema:** En este capítulo se muestra la implementación de la solución técnica. Para ello se utilizará todo lo descrito en capítulos anteriores, así como la experiencia técnica adquirida en el lenguaje Android. Se seguirá la arquitectura descrita, y se cumplirán los objetivos y requisitos fijados. Además, se describirá la metodología utilizada para el desarrollo del software.

5) **Evaluación y resultados:** En este capítulo se pondrá a prueba la aplicación desarrollada, comprobando el correcto funcionamiento de cada una de sus partes. Se analizarán los resultados obtenidos, y se concluirá si es correcta o no.

6) **Marco regulador:** En el desarrollo de este capítulo se realizará un repaso de la legislación vigente en España y en el marco de Europa, que afecta a este TFG. Se trata de poner de manifiesto las leyes de obligado cumplimiento, en caso de querer lanzar la aplicación al público general.

7) **Conclusión:** En este capítulo se presentarán las conclusiones extraídas de la realización y desarrollo de este TFG. Para ello se revisarán los objetivos alcanzados, y se establecerán una serie de líneas de trabajo futuras.

   Anexo A - Gestión del proyecto: Este anexo detallará la gestión del proyecto y la estimación de presupuesto. Se trata de un punto importante en el desarrollo del proyecto, ya que mostrará la gestión realizada, tanto de tiempo como de recursos. Para ello se detallarán las fases, se realizará un diagrama de Gantt, y por último se detallarán los costes de herramientas y personal.

   Anexo B - Manual de usuario: Este anexo es una breve guía de usuario. Está destinado a explicar el funcionamiento general del sistema.

   Anexo C - English versión: Este anexo, cumpliendo la normativa vigente para los alumnos del plan 2011, realizará una traducción al ingles de las siguientes partes:
   - Introducción completa.
   - Resumen del proyecto de entre 5 y 10 hojas.
   - Conclusiones completas.

   Para facilitar la lectura en inglés se ha determinado la realización de un resumen entre los capítulos 2 y 6. Esto unido a la traducción completa de la introducción y conclusiones permitirá contar con una versión reducida de este TFG en inglés.

2. Estado del arte

En este capítulo se realiza una revisión de las bases técnicas del TFG. Para ello se analizan las soluciones que existen actualmente en el mercado al problema de la congestión viaria, los dispositivos y las tecnologías disponibles, así como el estado en el desarrollo de aplicaciones móviles. Todo ello sirve como investigación que fundamenta las decisiones tomadas durante la realización del trabajo.

2.1. Soluciones/Aplicaciones actuales

Actualmente existen diferentes soluciones en forma de aplicación (APP) que ofrecen información del tráfico en tiempo real. Para recolectar esta información y poder ofrecérsela al usuario, cada una de ellas utiliza distintos recursos y/o algoritmos de detección. A continuación, se realiza un análisis de estas aplicaciones y sus algoritmos, lo cual nos ayuda a definir los aspectos diferenciales de nuestra solución.

2.1.1. Google Maps

Lanzado en 2005 por Google, surge como un servicio de mapas en la web para, posteriormente, pasar a ser tambien una aplicación móvil. En la actualidad presta servicios de localización, de búsqueda de direcciones, de navegación (en distintos medios de transporte), estado del tráfico (lanzado en 2007), etc.

Para la obtención de información sobre el estado del tráfico en las carreteras, Google cuenta principalmente con cuatro vías:

Sensores: En las primeras versiones tras su lanzamiento, Google Maps dependía completamente de los sensores instalados en carreteras o vehículos que circulaban por ellas.

Base de datos: Además, se utilizaba información basada en historial de tráfico, que gracias a los algoritmos de Google podía prever el nivel de congestión en las carreteras [5][6]. Aún hoy está disponible este historial de tráfico desde la web de Google Maps, permitiéndonos visualizar un histórico del tráfico para un día de la semana y hora concretos. Para la distinción del nivel de tráfico, Maps dibuja sobre la carretera un color que indica el nivel de congestión previsto.

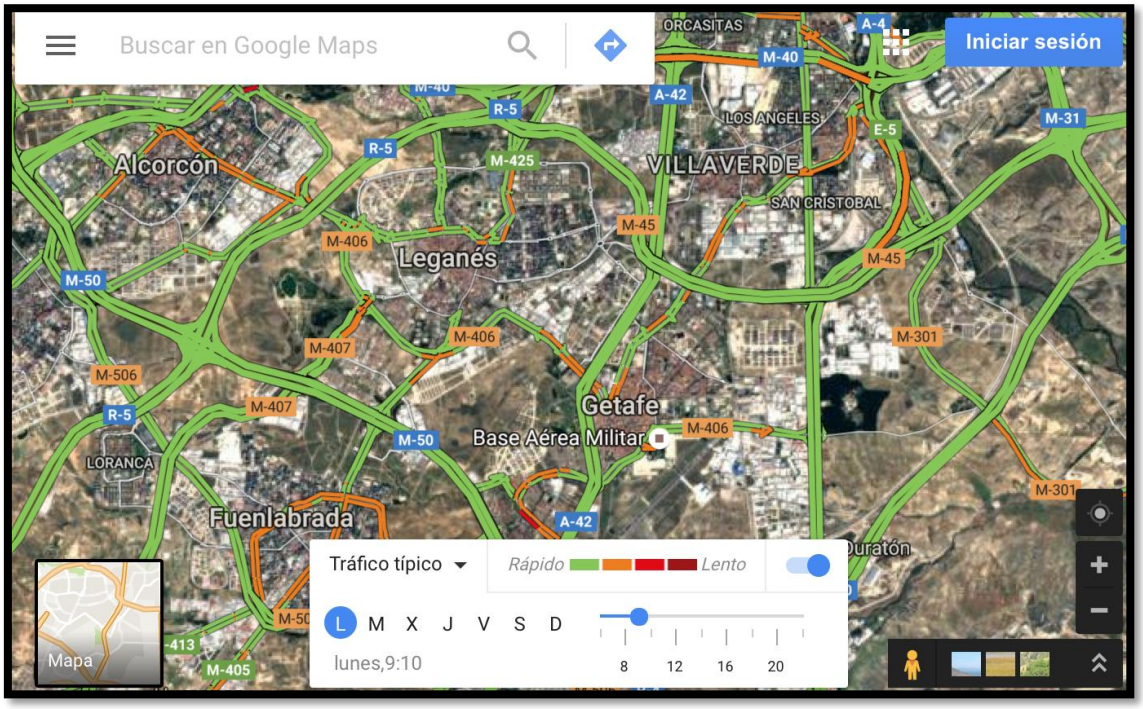

Figura 2.1: Vista de la predicción de tráfico basada en el histórico de datos. (Fuente Google Maps)

Usuarios: En 2007, con la adquisición de ZipDash's technology [34] por parte de Google, se integra un nuevo método de detección de información de tráfico, que permite en tiempo real determinar el estado de una carretera. Para ello, tal y como informa Dave Barth[1] en un artículo de agosto de 2009 [7], se utiliza la información recibida por todos aquellos usuarios de la aplicación con GPS y datos activados.

Es decir, la ubicación del usuario junto con otros datos como velocidad, altitud o posición, es enviada continuamente a los servidores de Google. Gracias a la combinación de los datos obtenidos de diferentes Smartphone, y su comparación con la velocidad máxima de la vía, se puede determinar el estado de congestión de la carretera. De esta forma el usuario actúa como sensor que envía datos, que Google se encarga de analizar y actualizar en tiempo real.

Gracias a este análisis del tráfico en tiempo real, Google Maps es capaz de ofrecer el tiempo aproximado en rutas de navegación, o rutas alternativas más rápidas en caso de congestión en nuestra ruta original.

Información de terceros: Acuerdos con distintos organismos encargados del tráfico a todos los niveles de la administración pública,

[1] Manager de Google Maps desde octubre 2012 - hasta la actualidad.

así como adquisición de empresas privadas, proveen a Google Maps de información del tráfico. Un ejemplo de ello es Waze [8], adquirida por Google en 2013. Waze es una aplicación colaborativa, cuyo algoritmo de detección será tratado en sucesivas secciones.

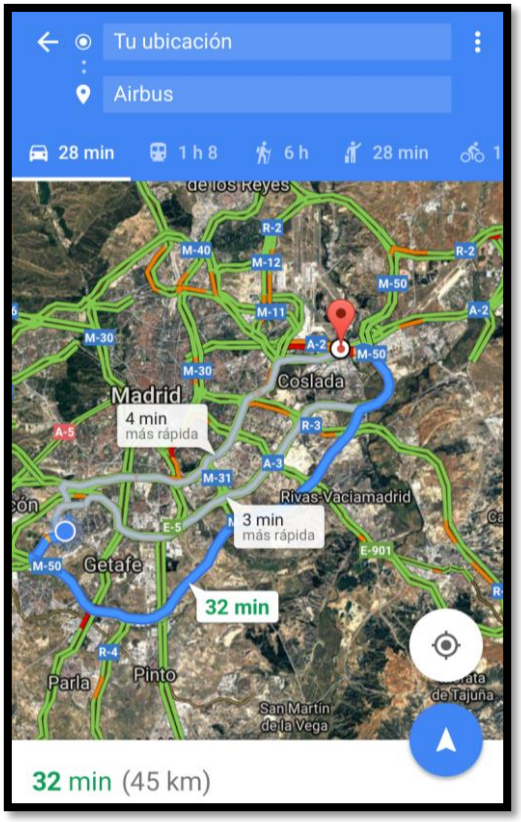

Figura 2.2: Tráfico en tiempo real, y tiempo estimado de ruta. (Fuente Google Maps APP)

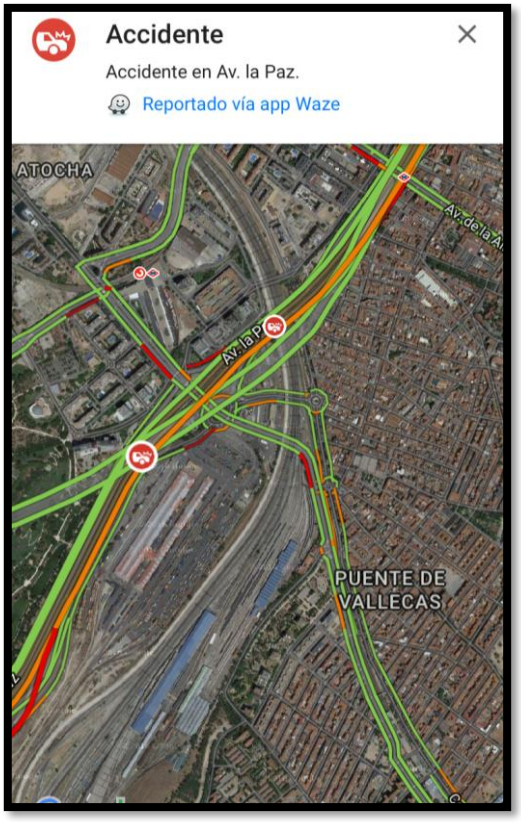

Figura 2.3: Información de terceros proporcionada por Google Maps. (Fuente Google Maps APP)

2.1.2. Dirección General de Tráfico

Como organismo nacional, del estado español, se encarga de la formación de los usuarios de las vías, y de asegurar la circulación de vehículos con seguridad y fluidez. La DGT [2] cuenta con diversas herramientas para la medición y detección del estado del tráfico.

Para conocer e informar sobre el estado de la circulación, la DGT cuenta con diversos recursos. Un ejemplo de ello son los cientos de cámaras y sensores repartidos por las principales vías de España, a través de los que se puede detectar la fluidez en la circulación.

Figura 2.4: Cámara correspondiente a la carretera M-50 KM 79.8 sentido decreciente. (Fuente DGT)

Además, la DGT también cuenta con diversos mapas que muestran: el estado del tráfico, incidencias en la circulación, previsiones de intervención planificadas en la red viaria, así como listados de puntos negros y posibles restricciones especiales a determinados tipos de vehículo.

Figura 2.5: Mapa de incidencias de circulación en tiempo real. (Fuente DGT)

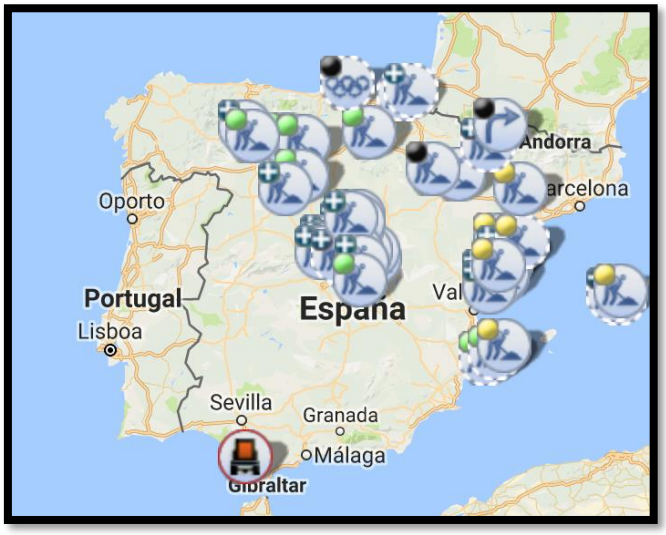

Figura 2.6: Mapa de intervenciones y restricciones planificadas. (Fuente DGT)

Toda esta información es visible a través de su página web, o de su aplicación para dispositivos móviles [35]. Además, la DGT se coordina con otras agencias estatales para la obtención de otro tipo de informaciones, que puedan afectar al transcurso del tráfico, como por ejemplo la información meteorológica provista por la Agencia Estatal de Meteorología (AEMET).

2.1.3. Aplicaciones colaborativas: Waze

Fundada en 2008, Waze [8] es una aplicación social que permite a sus usuarios la navegación asistida por GPS y la detección del tráfico en tiempo real. Se basa en la premisa de ser una aplicación mantenida por el usuario y que aprende del mismo.

Waze pertenece a una nueva generación de aplicaciones llamadas colaborativas. Estas son aplicaciones interactivas, que buscan un intercambio activo del usuario para captar información del mismo, con el fin de mejorar la aplicación.

En el caso de Waze, la interacción se efectúa mediante el establecimiento de avisos por parte del usuario. Durante la navegación hacia su destino, el usuario cuenta con un apartado de alertas por diferentes causas, entre ellas encontramos:

- Tráfico: Puede ser moderado, denso o detenido.
- Policía: Puede ser visible o no visible.
- Accidente: Puede ser leve o grave.
- Peligro: Puede ser en la vía, en el arcén o climatológico.
- Cierres: Por cierre de una vía o calle.

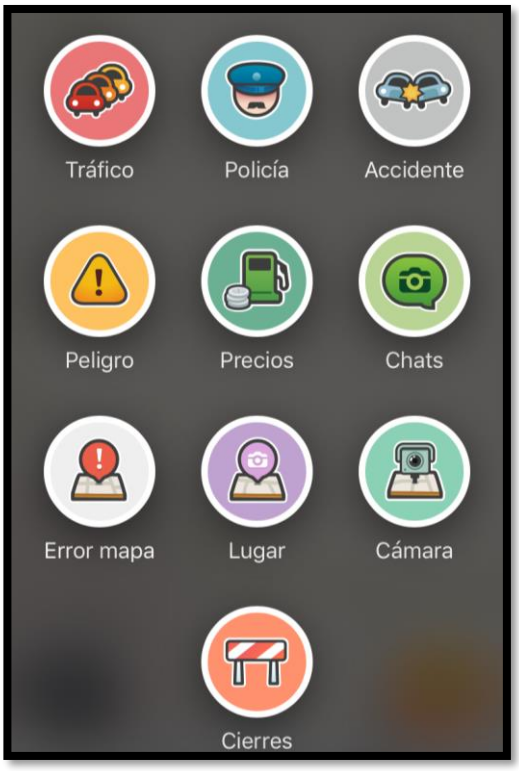

Figura 2.7: Distintos tipos de alertas. Disponibles para el aviso de eventos por parte de un usuario. (Fuente Waze)

El usuario únicamente deberá pulsar la alerta correspondiente, y una notificación será enviada a los servidores de Waze. A partir de ese momento, y gracias a diferentes algoritmos que procesan el número de alertas de un suceso, el número de usuarios en el lugar y otros datos, los servidores son capaces de determinar si existe algún tipo de congestión en la vía. Una vez determinado el alcance del suceso, dicha alerta podrá ser enviada a otros usuarios Waze. Estos datos podrán ser tenidos en cuenta en la elección de su ruta de navegación más óptima.

Figura 2.8: Detalle de alertas informadas, que se encuentran en la actual ruta del usuario hacia su destino. (Fuente Waze)

Figura 2.9: Detalle de la obtención de rutas, teniendo en cuenta las alertas de la vía. (Fuente Waze)

2.1.4. Trabajos de investigación

Previo a la realización de este proyecto, se han investigado diferentes soluciones alternativas. Todas ellas buscan dar una solución al problema de la congestión viaria, y lo realizan mediante la investigación de nuevas técnicas. A continuación, realizaré un breve resumen de los artículos más destacados, y de la solución que proponen:

TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones[27]**:** La solución implementada en este TFG se basa principalmente en este artículo. Se caracteriza por incorporar un

punto de vista, donde los dispositivos móviles son protagonistas. Se trata de utilizar estos dispositivos y sus sensores para la captura de datos, y su posterior detección de eventos. El dispositivo con mayor importancia es el acelerómetro, capaz de detectar eventos de frenado y aceleración. En cuanto al uso del acelerómetro, este artículo propone una metodología para su reorientación y la captura de aceleraciones en los diferentes ejes. Esta metodología será utilizada en el desarrollo de la aplicación, y se explicará en sucesivos capítulos.

A Sensor Fusion Method for Smartphone Orientation Estimation[31]**:** Este artículo no propone una solución al problema de la congestión viaria como tal. Sin embargo, como ocurre en el artículo anterior, propone un método para la orientación del acelerómetro de un Smartphone. Puesto que la reorientación de este sensor, es fundamental para la detección de eventos de aceleración/frenado, este artículo siembra las bases de nuestro proyecto. Los dispositivos utilizados en él son: el magnetómetro y el giroscopio.

Video Based Vehicle Detection and Its Application in Intelligent Transportation System[37]**:** En este artículo del departamento de ingeniería eléctrica y de computadores de la Universidad de Nevada, se analiza un sistema de detección de vehículos inteligente, basado en video. La idea principal es capturar mediante cámaras, imágenes o videos, el estado de las carreteras. Se automatizará el mecanismo de análisis para poder detectar distintos patrones, que indiquen eventos en las carreteras.

Android Smartphone Application for Driving Style Recognition[36]**:** Se trata de un proyecto de fin de carrera. En él, su autor utiliza la fusión de los sensores acelerómetro, giroscopio, y magnetómetro, incorporados en los dispositivos móviles, para tratar de definir distintos patrones de conducción. De esta manera será capaz de dar consejos de conducción.

2.1.5. Conclusiones sobre las soluciones analizadas

Tras realizar un análisis de las principales soluciones del mercado, y los principales artículos que proponen soluciones alternativas, podemos concluir:

- Que las soluciones existentes en el mercado, no cumplen ninguno de los requisitos impuestos para este proyecto. En el caso de Google, la privacidad queda en entredicho por el envío continuo de datos, por no efectuarse una detección de forma local. En el caso de Waze, tampoco tenemos una detección local de eventos, y el sistema puede no ser cómodo para el usuario, al tener que avisar él mismo de posibles eventos. En cuanto a la DGT, podemos concluir, que sin duda cuentan

con grandes recursos para la detección de eventos, si bien su sistema de avisos al usuario no saca provecho de los mismos.

- Que existe un mercado potente en torno a la navegación GPS, con grandes carencias en la protección de la privacidad. Por lo que comercialmente, un detector de eventos de congestión de forma local, que evite el envío continuo de datos a servidores, tiene un gran nicho de mercado.
- Que los artículos de investigación proponen soluciones alternativas incompletas. Esto es así, puesto que se detectan eventos de frenado y aceleración, pero no se utilizan estos datos para definir la causa de los mismos.
- Que los artículos de investigación proponen una metodología común para la reorientación del acelerómetro. Esta metodología presenta un funcionamiento satisfactorio, y posibilitará, en nuestro proyecto, el uso del acelerómetro como sensor principal para la detección de eventos.

2.2. Dispositivos móviles.

El desarrollo tecnológico vivido en los últimos años ha propiciado la aparición de dispositivos móviles, con grandes capacidades de cómputo y numerosos sensores. Los llamados Smartphone surgen en la primera década del siglo XXI. En 1999, la compañía japonesa NTT DoCoMo [38] fabricó el primer Smartphone, capaz de lograr una gran tasa de adopción entre los habitantes de un país. Se denomina Smartphone a los teléfonos inteligentes, con gran capacidad para el manejo de datos, mayor conectividad y mejores prestaciones que los teléfonos móviles convencionales.

Gracias al avance de la tecnología, y lo aprendido con los Smartphone, se han desarrollado distintos dispositivos tales como: *Tablet, Smartwatch*, e infinidad de dispositivos electrónicos. Nos encontramos en la era del *always on*, es decir, siempre conectados. Todos estos dispositivos electrónicos tienen algo en común: su gran adopción por parte de los usuarios y su gran capacidad para la obtención y manejo de datos de diferentes sensores (similar a una computadora en miniatura).

Lugar	**España**	**Singapur**	**Italia**	**Alemania**	**Hong Kong**	**Corea del sur**
Penetración (%)	87	85	84	84	82	82

Tabla 2-1: Top 6 de los lugares con mayor tasa de dispositivos móviles, en comparación con su número de población (2014). (Fuente WeAreSocial)

En el mercado podemos encontrar dispositivos que cuentan con conectividad a internet, ya sea mediante Wi-Fi o internet móvil (2G, 3G, 4G), sistema de posicionamiento global o GPS para la detección de la ubicación del dispositivo, acelerómetros y brújulas capaces de detectar movimientos y aceleraciones. A continuación, se muestra una breve descripción de aquellos sensores estudiados para la toma de datos y detección de eventos, que permitan solucionar de forma alternativa el problema planteado durante este TFG.

2.2.1. Sistema de localización: GPS

Sistema de posicionamiento global[10], que funciona mediante una red de 24 satélites situados en la órbita del planeta Tierra. Su exactitud permite determinar la posición de un objeto con una precisión del orden de centímetros, en cualquier lugar del planeta.

Para determinar la posición de dicho objeto, el receptor se encarga de localizar al menos 3 satélites de la red. Una vez localizados se envía una señal a cada uno de ellos, sincronizando así el reloj del GPS, para posteriormente medir el retardo entre las distintas señales. Este retardo permite conocer la distancia a cada uno de los satélites. pudiendo así triangular el lugar exacto donde se sitúa el objeto.

A pesar de la existencia de otros sistemas de navegación (Galileo desarrollado por la UE o GLONASS gestionado por Rusia), GPS es el más extendido. Su generalización en el mundo de los dispositivos móviles ha permitido el desarrollo de diferentes aplicaciones. La mayoría de estos dispositivos cuentan con un chip GPS, que permite a los desarrolladores utilizar la ubicación en sus aplicaciones, lo cual incrementa las opciones durante el proceso de desarrollo.

2.2.2. Sensores

Los dispositivos móviles cuentan con diferentes sensores que hacen posible detectar distintos tipos de datos. A continuación, una breve explicación de algunos de ellos:

Acelerómetro[11]**:** Es todo aquel sistema que tiene la capacidad de medir aceleraciones. En la actualidad, aquellos integrados en dispositivos móviles, se construyen con chips de silicio de tres ejes (X, Y, Z), incluyendo la electrónica capaz de procesar las señales producidas por el acelerómetro.

Con la orientación apropiada, y el desarrollo de software adecuado, un acelerómetro puede ser utilizado para diversas utilidades tales como:

medición de la actividad física, así como de las calorías quemadas al captar el movimiento del usuario, orientación de la pantalla basada en el movimiento del dispositivo, juegos que detectan el movimiento como parte de su manejo, etc. Cabe destacar, que los dispositivos actuales de alta gama presentan acelerómetros caracterizados por su bajo consumo de batería, de cara a la medición de actividad física durante el día.

Barómetro[39]**:** Es un instrumento que mide la presión atmosférica. La presión atmosférica es el peso por unidad de superficie ejercida por la atmósfera. En la actualidad algunos Smartphone de gama alta cuentan con este tipo de sensor para tareas, como medir el número de pisos subidos, o la presión atmosférica para pronosticar el tiempo.

Magnetómetro[40]**:** Es un instrumento que cuantifica en fuerza o dirección la señal magnética de una muestra. Se utiliza en los Smartpone para detectar el polo norte magnético, o para detectar imanes de fundas inteligente (aquellas que apagan y encienden la pantalla de forma automática).

Fotoeléctrico [41]**:** Es un dispositivo electrónico que responde al cambio en la intensidad de la luz. Es de utilidad para adaptar el brillo de la pantalla a las condiciones lumínicas del momento.

Además de los aquí mencionados, los dispositivos móviles cuentan con: Sensores de huella, de ritmo cardiaco, giroscopios, de humedad, etc.

2.2.3. Conectividad: Redes inalámbricas y móviles

El Internet móvil es posible gracias a las denominadas redes inalámbricas y móviles[12]. Podemos diferenciar, de forma general, dos tipos de conexiones para la obtención de internet móvil: redes Wi-Fi y redes de Banda Ancha Móvil (BAM).

Las redes Wi-Fi permiten la conexión inalámbrica de dispositivos móviles, a través de un punto de acceso de red. El alcance es de aproximadamente 20 metros.

Por otro lado, las redes de BAM, funcionan a partir de la cobertura móvil. Según la tecnología que implementen pueden ofrecer distintos anchos de banda (GPRS, 3G, 4G).

Ambas tecnologías son ampliamente utilizadas, tanto, que en los últimos años el crecimiento de líneas fijas con acceso a Banda Ancha Fija (potenciales

de ofrecer Wi-Fi), y móviles con acceso a Banda Ancha Móvil, ha sido desde 2010 hasta 2014 bastante notable.

A continuación, los datos de penetración en España [13] para ambos servicios, entre 2010 y 2014. Se ha de tener en cuenta que se toma como referencia la cifra de población total, que en enero de 2014 ascendía a 46.512.199 habitantes en España.

(líneas/100 habitantes)

Año	Penetración
2010	22,9
2011	23,9
2012	24,6
2013	26,2
2014	28,0

Tabla 2-2: Tasa de penetración de la banda ancha fija sobre la población. (Fuente CNMC)

(líneas/100 habitantes)

Año	Penetración
2010	23,7
2011	37,1
2012	52,9
2013	67,0
2014	78,3

Tabla 2-3: Tasa de penetración de banda ancha móvil sobre la población. (CNMC)

Además, en función de la tecnología utilizada, se distingue entre diferentes tipos de conexiones móviles y Wi-Fi. A continuación, una tabla aclaratoria sobre las características para cada tipo de conexión.

Estándar Wi-Fi	Velocidad máxima	Banda de Frecuencia
802.11	Entre 1 y 2 Mbps	2.4 Ghz
802.11a	54 Mbps	5.15 Ghz
802.11b	11 Mbps	2.4 Ghz
802.11g	54 Mbps	2.4 Ghz
802.11n	> 100 Mbps	2.4 Ghz – 5 Ghz

Tabla 2-4: Características estándares de la conexión Wi-Fi.

Tecnología	Velocidad promedio de bajada	Velocidad promedio de subida
GSM (2G)	1.8 kB/s	1.8 kB/s
GPRS (2.5G)	7.2 kB/s	3.6 kB/s
EDGE (2.75G)	29.6 kB/s	29.6 kB/s
UMTS (3G)	48 kB/s	48 kB/s
EDGE	59.2 kB/s – 237 kB/s	59.2 kB/s – 118 kB/s
HSPA (3.5G)	1.706 kB/s – 5.25 MB/s	720 kB/s – 1.437 MB/s
LTE	21.625 MB/s – 40.750 MB/s	7.25 MB/s – 10.750 MB/s

Tabla 2-5: Características de la tecnología de redes móviles.

2.2.4. Captura de imágenes

En la actualidad, los Smartphone cuentan con las capacidades de cámaras fotográficas y de video. Esto les permite capturar imágenes,

fotografías y videos de diferentes calidades y características. A día de hoy, los Smartphone de gama alta, cuentan incluso con grabación a cámara lenta, zoom digital, estabilizadores ópticos, etc.

2.2.5. Conclusiones sobre los dispositivos móviles

Tras realizar un análisis sobre los dispositivos móviles, y el estado de la tecnología que incorporan, podemos concluir:

- Que la tecnología móvil y en concreto los Smartphone, tiene una gran acogida en la sociedad, contando en nuestro país con la mayor tasa de penetración.
- Que la tecnología incorporada en los dispositivos móviles está muy desarrollada. Los Smartphone cuentan con numerosos sensores, que les permitirán interaccionar con el medio y recabar información.
- Que el gran desarrollo del acelerómetro, por su precisión, y su bajo consumo de batería, permite que sea la opción elegida para este proyecto, por delante de otras opciones como: La captación de imágenes, y su posterior análisis, o el uso intensivo del GPS.

2.3. Desarrollo de software en dispositivos móviles

Aunque no hay una fecha exacta para datar el desarrollo de las primeras aplicaciones en dispositivos móviles, sí que se puede determinar que su incremento e importancia creció con la aparición de los llamados Smartphone. Se denomina aplicación móvil o APP, al software desarrollado y diseñado para ser ejecutado en dispositivos móviles, tales como Smartphone, Tablet o todo tipo de *gadget* informático.

Las APP son programas diseñados para facilitar al usuario realizar una tarea, ya sea de ocio, educativa, trabajo, acceso a diferentes servicios, etc. Este tipo de aplicaciones suelen estar ubicadas en los dispositivos, y están escritas en diferentes lenguajes de programación según su plataforma.

En las próximas secciones, se realizará una introducción a los diferentes sistemas operativos existentes en el mercado del software para dispositivos móviles, para posteriormente centrarse en Android (el SO seleccionado para el desarrollo de este TFG) y algunas herramientas de software adicional que han resultado interesantes durante el desarrollo de este proyecto.

2.3.1. Introducción al desarrollo en dispositivos móviles

Para entender la magnitud del mercado del desarrollo en dispositivos móviles, es importante conocer los diferentes SO con mayor cuota de mercado en la actualidad.

Android[14]**:** Lanzado al público por primera vez en septiembre de 2008, surge como un SO móvil basado en el núcleo de Linux y creado como software libre, tanto para usuarios como para desarrolladores. La principal meta de este SO es mejorar la experiencia de los usuarios. Comprado en 2007 por Google, en la actualidad Android es uno de los agentes más importantes del mercado de los dispositivos móviles.

iOS[15]**:** Desarrollado por Apple, sale por primera vez al público en 2007 con el lanzamiento del primer iPhone. Se caracteriza desde su primera versión por la simplicidad en su manejo, haciendo que su interfaz funcione con gran fluidez. Entre sus puntos fuertes, iOS presume de ser desarrollado por la mayor empresa, en términos de capitalización bursátil, del mundo.

Existen además otros SO en el mercado de los dispositivos móviles, Windows Phone de Microsoft, o BlackBerry 6 de BlackBerry, pero cuya implantación es mínima. A continuación, podemos ver una estadística de las ventas de Smartphone, clasificada por SO, durante Diciembre de 2015, en algunos de los mercados mundiales más relevantes[16].

	España	Francia	EEUU	Japón	China
Android	86.3%	69.9%	59.1%	44.4%	71.4%
BlackBerry	0.0%	0.6%	0.1%	1.2%	0.0%
iOS	12.2%	20.5%	39.1%	54.1%	27.1%
Windows	1.5%	8.7%	1.6%	0.0%	1.2%
Otros	0.0%	0.4%	0.1%	0.3%	0.2%

Tabla 2-6: Cuota de ventas del mercado de Smartphone, durante diciembre de 2015. (Fuente Kantar World Planet)

2.3.2. Sistema Operativo Android

Basado en un núcleo Linux, y desarrollado por Google, Android es sin duda una de las mejores opciones para el desarrollo de software móvil. Si algo caracteriza a este SO es su definición como software libre, ya que uno de los objetivos de Android es llegar al mayor número, tanto de usuarios como de desarrolladores.

Podemos destacar a Android por su compatibilidad entre distintos dispositivos. Por tratarse de un proyecto de software libre, es una plataforma a la cual cualquier fabricante de hardware puede acceder. Esto ayuda a los desarrolladores, ya que al lanzar sus aplicaciones no necesitan preocuparse de la compatibilidad con distintos dispositivos hardware.

Las APP Android están escritas en el lenguaje de programación Java. Gracias al SDK de Android y sus herramientas, se puede compilar el código generado de la APP. El SO operativo cuenta con una máquina virtual cuyas funciones son la gestión de memoria y recursos de forma eficiente.

Para su compatibilidad con distintos dispositivos, y su definición como código abierto basado en software de Linux, Android cuenta con la siguiente arquitectura de componentes[17]:

Kernel de Linux: Es el núcleo del SO. Son los cimientos de Android sobre los que se gestionan recursos como la memoria, la capa de protocolos, los procesos y los drivers o controladores de dispositivos.

Hardware abstraction layer: Capa de abstracción del hardware. Provee una interfaz estándar para diferentes componentes hardware, como puede ser la cámara o el módulo Bluetooth.

Android Runtime: Se trata del entorno de ejecución. Cada aplicación lanza su propia instancia de Android Runtime (ART). Las características de ART principales son:

- Capacidad para la compilación antes de la ejecución, o en tiempo de ejecución.
- Optimización del recolector de basura (que se encarga de gestionar la memoria).
- Soporte para la puesta a punto, con herramientas de diagnóstico de software.

Bibliotecas: Nativas de C/C++. Se utilizan por distintos componentes del sistema. Entre sus librerías se encuentra la del sistema C, librerías multimedia para la reproducción y grabación de formatos de audio y sonido, gestores de contenidos visuales tanto en 2D como en 3D, motores de navegación web o el motor de BBDD relacionales SQLite.

Framework de aplicaciones: Son APIs de Java. Basado en APIs de código en lenguaje Java, es la puerta de acceso a todas las características disponibles en el SO Android. Estás APIs son como bloques, que proveen lo necesario para el desarrollo de APPs,

simplificando el uso de los diferentes componentes o servicios. Entre ellos:

- Un sistema de interfaz gráfica, que incluye listas, *grids*, cajas de texto o botones para la construcción de la interfaz de usuario.
- Un manager de recursos, que provee acceso a recursos fuera del código como, textos, gráficos y ficheros.
- Un manager de actividades, que maneja el ciclo de vida de las aplicaciones y provee una pila de navegación común.

Aplicaciones: Android cuenta con un núcleo de aplicaciones por defecto como email, SMS, calendarios, navegador, contactos y otras. Además, el usuario podrá instalar las aplicaciones que desee, las cuales estarán basadas en los servicios provistos por capas inferiores.

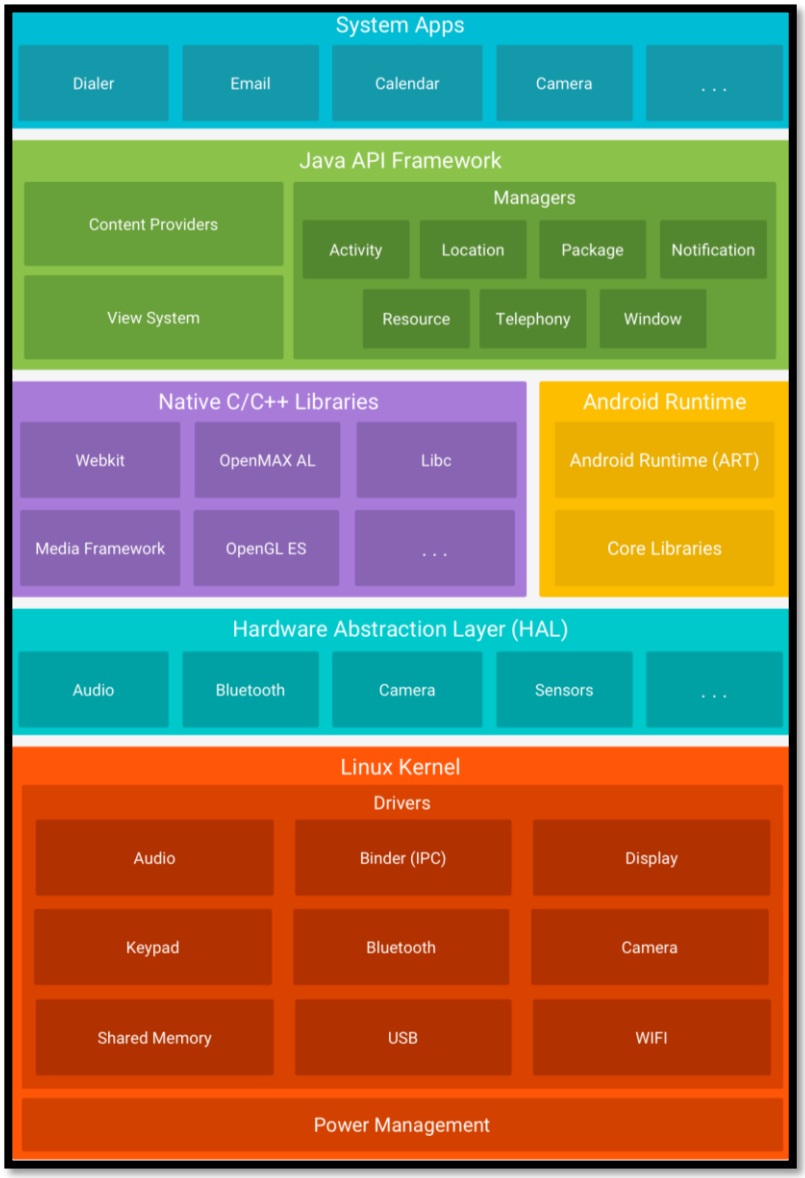

Figura 2.10: Pila de capas software por las que está formado Android. (Fuente Web desarrolladores Android)

Durante el desarrollo de una aplicación en Android se debe tener en cuenta su gestión automática de procesos. Existen tres niveles que determinan la prioridad de un proceso, e indican al SO si debe eliminarse o no.

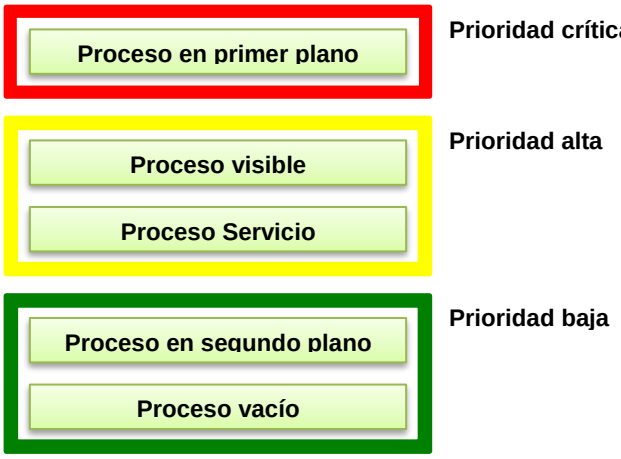

Figura 2.11: Jerarquía de procesos en Android.

Existe una serie de elementos, que son clave, por resultar imprescindibles para el desarrollo en Android. Son cuatro: Activity, Service, Content Provider y Broadcast Receiver.

- **Service:** Permite llevar a cabo operaciones en segundo plano o background y, por tanto, no proporcionan al usuario una interfaz.
- **Content Provider:** Proporciona facilidades de acceso a datos estructurados. Además de encapsular los datos, proporcionan mecanismos para definir la seguridad de los datos.
- **Broadcast Receiver:** Permiten que se reciban notificaciones y otros datos, desde ciertos componentes. El componente que envía la notificación lo hacen a través de un *sendbroadcast*. Los componentes que la reciben se asocian o suscriben a un *Receiver*.
- **Activity:** Es el componente básico que permite crear módulos que interaccionan con el usuario. Cada *activity* tiene asociada una vista. La vista esta constituida por un conjunto de elementos gráficos.

Además de estos componentes, cabe destacar los *intent,* que permiten el envío de mensajes entre componentes y el *manifest*, que es un archivo de configuración donde podemos aplicar las configuraciones básicas de una aplicación.

Una aplicación está formada por uno o varios componentes y siempre al menos por un *activity*. Una vez lanzada la aplicación, esta pasará por diferentes estados dependiendo del uso que se le esté dando. Esta serie de estados conforman el llamado ciclo de vida de la APP [17].

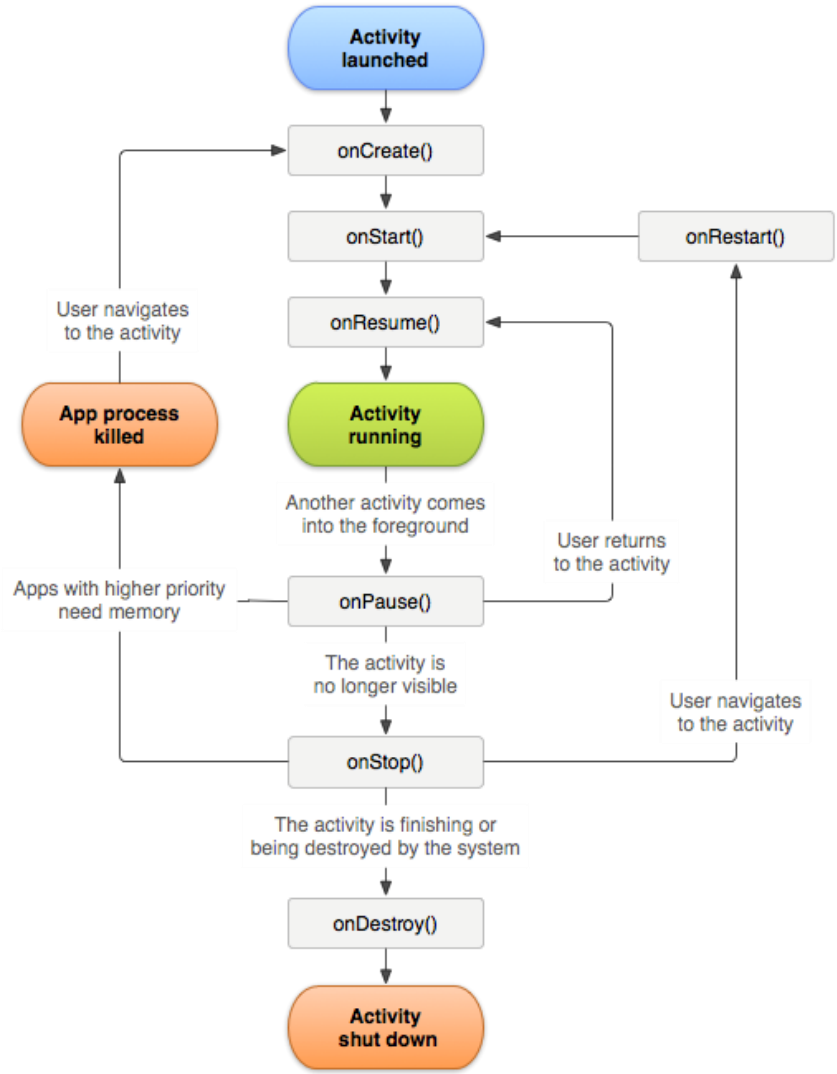

Figura 2.12: Ciclo de vida de una aplicación Android. (Fuente Web desarrolladores Android)

Se pueden controlar los diferentes estados gracias a una serie de métodos predefinidos, donde el usuario podrá añadir los comportamientos deseados en su aplicación. A continuación, una breve explicación de los métodos y su uso.

- **`onCreate()`:** Implementa las acciones a realizar cuando se crea la aplicación.
- **`onStart()`:** Implementa las acciones a realizar cuando la aplicación pasa a primer plano.
- **`onResume()`:** Implementa las acciones a realizar cuando la aplicación pasa a primer plano, tras haber estado en pausa.
- **`onStop()`:** Implementa las acciones a realizar cuando se para la aplicación.
- **`onRestart()`:** Implementa las acciones a realizar cuando se reinicia la aplicación, tras haber sido parada.

- **`onPause()`:** Implementa las acciones a realizar cuando se pausa la aplicación.
- **`onDestroy()`:** Implementa las acciones a realizar cuando se cierra la aplicación.

2.3.3. Librerías de datos sobre tráfico

Como parte de su apuesta por el desarrollo de software libre, Android incluye la posibilidad de utilizar e implementar librerías de terceros. Aunque los datos proporcionados por Android, mediante el acceso al GPS del dispositivo móvil, son bastante amplios (incluyendo coordenadas de localización, altura, velocidad, etc.), en ocasiones es necesario conocer otro tipo de datos ajenos al dispositivo. Es el caso de los datos sobre tráfico.

Durante el desarrollo de este TFG, surgirá la necesidad de conocer datos como la velocidad máxima de la vía por la que se circula. Para ello será necesario el uso de librerías de datos sobre tráfico de terceros, pasamos a analizar algunas de ellas.

Google API[18]**:** Desarrollada por Google, provee una interfaz de programación para aplicaciones, con acceso a los servicios provistos por esta empresa. Para el problema que nos ocupa, el análisis de la congestión del tráfico, la API de Google nos proporciona algunas de las siguientes herramientas:

- Capa de datos de tráfico.
- Mapas.
- Dibujado y marcado de mapas.
- Eventos.

HERE WeGo API[19]**:** Creada por Nokia, en la actualidad su propietaria es un consorcio formado por empresas potentes del sector automovilístico (como Audi o BMW). Con su API, provee servicios relacionados con mapas, cartografía y estado del tráfico. Una de las características más interesantes es su base de datos de límites de velocidad en diferentes vías.

Para la utilización de sus servicios, HERE WeGo ofrece una conexión con los datos necesarios a través del protocolo HTTP. La respuesta a esta petición contendrá los datos, en formatos XML o JSON. Android cuenta con métodos implementados en su API para el tratamiento de ambos lenguajes, lo cual simplifica la tarea de implementación de esta API de Here WeGo. [19]

2.3.4. Conclusiones sobre el desarrollo de software en dispositivos móviles

Tras realizar un análisis sobre el estado del desarrollo de software, en dispositivos móviles, podemos concluir:

- Que existen varios SO potentes en el mercado, con grandes herramientas para los desarrolladores, pero destaca fuertemente la posición de Android, por ser líder en número de dispositivos compatibles y el actor con mayor número de usuarios.
- Que existen librerías de terceros, compatibles con Android, y que permiten la implementación de aquellas características, que este SO no tiene por sí mismo. Algunas de estas proporcionan información sobre tráfico que nos será útil durante el desarrollo de este proyecto.

2.4. Herramientas de cálculo: Matlab

Este TFG centra su solución en la toma de datos con el acelerómetro y, en menor medida, con el GPS y una librería de terceros. Posteriormente se analizarán de forma local estos datos, para establecer umbrales que nos permitan la detección de eventos. Previo al establecimiento de estos umbrales, será necesario estudiar los datos, y aquí es donde entra en juego Matlab y sus potentes características.

Matlab [20] es una herramienta de software multiplataforma (Linux, Windows, MacOS), con un IDE y lenguaje de programación propio. Está optimizado para la resolución de cálculos y problemas de ingeniería. Entre sus prestaciones encontramos:

- Manipulación de matrices.
- Representación de datos y funciones.
- Implementación de algoritmos.
- Creación de interfaces de usuario.
- Comunicación con otros dispositivos, o lenguajes.

Especialmente útil para la manipulación de datos, y la representación de los mismos, permite dibujar gráficas en 2D y 3D. Además, Matlab posee su propia API con funciones definidas, lo cual facilita la tarea del desarrollador.

3. Análisis y diseño de la solución técnica

En este capítulo se analiza y se describe el diseño de la solución técnica. Se realiza a través de la descripción de la arquitectura para, a continuación, presentar los requisitos a resolver y los casos de uso a tratar.

Una vez definido el tema de este proyecto, la obtención de información sobre tráfico en las carreteras, es necesario definir una solución técnica eficaz, que cubra unas necesidades y expectativas que le permitan diferenciarse de otras soluciones. Se define como eje principal de la APP, el desarrollo de un algoritmo de detección de congestión del tráfico alternativo, cuyas características permitan:

- Una mayor protección de la privacidad del usuario.
- Un menor consumo de recursos.
- Una eficacia probada, similar a la ofrecida por los recursos ya existentes en el mercado.

3.1. Arquitectura

Para una mayor comprensión sobre la arquitectura de la APP, se ha decidido realizar una división en capas.

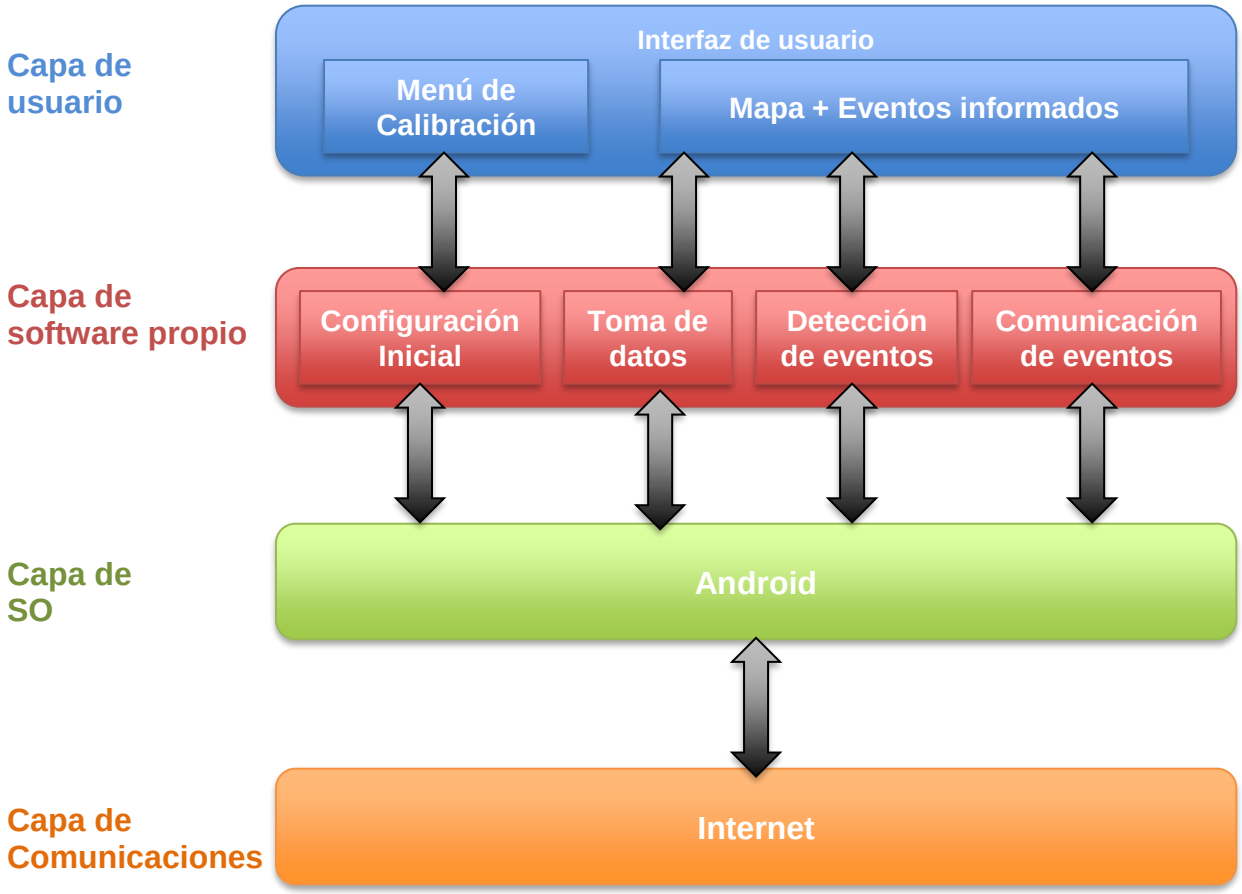

Figura 3.1: Arquitectura de la APP.

Como se observa en la figura 3.1, la arquitectura consta de 4 capas, que interactúan entre ellas de forma jerárquica. Se decide una división en diferentes capas, ya que es una forma modulable que permite la fácil sustitución de una de las capas. Por ejemplo, esto ocurriría si decidiésemos utilizar otra capa para la comunicación en lugar de internet, o si sustituyésemos Android por otro SO móvil. A continuación, una breve descripción de estas capas, sus componentes, y su función dentro de la arquitectura del sistema.

Capa de usuario: Su componente principal es la interfaz de usuario. Provee al usuario de un medio para comunicarse con las distintas funcionalidades del sistema, de una forma intuitiva y clara. Es un medio de entrada y salida, ya que puede tanto permitir la introducción de datos por parte del usuario, como mostrar los mismos. Debido a la simplicidad de esta APP, esta capa de usuario se divide principalmente en dos funcionalidades:

- Menú de Calibración: Permite la calibración y puesta a punto de todos los sensores para el inicio de la aplicación. Esta puesta a punto, será realizada de forma automática con ayuda del usuario, o bien, mediante la carga de configuraciones anteriormente guardadas.
- Mapa + Eventos informados: Permite informar al usuario de aquellos eventos ocurridos durante su navegación. Estos serán principalmente eventos de congestión en el tráfico.

Capa de software propio: Es aquella referida a los componentes software que han sido desarrollados para esta APP. Se apoyan en los servicios proporcionados por el SO Android, para la realización de diferentes tareas:

- Configuración inicial: Tiene principalmente dos funciones. Por un lado, cargar una configuración anteriormente guardada. Por otro lado, es capaz de configurar los sensores necesarios para la utilización de esta APP. Algunos de estos sensores son el GPS o el sensor de movimiento (acelerómetro), que deberá ser correctamente calibrado como veremos en el capítulo 5.
- Toma de datos: El sistema tomará de forma continuada medidas del acelerómetro. Además, en ocasiones señaladas, se podrá hacer uso del sistema GPS para la ubicación de un evento.
- Detección de eventos: Utiliza los datos obtenidos en la toma de datos, y con unos umbrales previamente establecidos, es capaz de distinguir eventos de congestión en la vía. Si esto ocurre, decidirá la existencia del evento o no, y guardará los datos necesarios para su posterior ubicación y envío.
- Comunicación de eventos: Si se detecta un evento, este podrá ser comunicado mediante el envío de datos a un servidor. Es

decir, si por ejemplo nuestro algoritmo de detección detecta congestión en un punto de la vía, este procederá a informar a un servidor de datos.

Es importante aclarar, que la comunicación de datos únicamente se producirá al detectar un evento de forma local en nuestro dispositivo. La configuración inicial pertenece a la fase de calibración y el resto pertenece a la fase de detección de la que hablaremos en el capítulo 5.

Capa de SO: En nuestro caso se trata del SO para dispositivos móviles Android. Como ya vimos durante el capítulo 2, este provee de herramientas para facilitar el acceso a los servicios proporcionados, por lo que mediará entre el hardware y nuestra solución software para la obtención de datos.

Capa de comunicaciones: Aunque no está actualmente implementado, se prevé la comunicación de eventos mediante la red a un servidor. Esto permitirá el aviso a otros usuarios de eventos de congestión en la vía, posibilitando el ofrecimiento de rutas alternativas.

Cabe destacar que la importancia de este TFG radica en su algoritmo alternativo de detección de eventos de tráfico, en concreto de congestión de vías. Por lo que su desarrollo se ha realizado con vista a una futura integración en sistemas de información de tráfico. Se trata de crear un sistema para la compartición de información entre usuarios, que avise de eventos, y permita avisar de estos también.

3.2. Requisitos

La especificación de requisitos de software (ERS) consiste en una descripción del comportamiento del sistema desarrollado. Para la descripción de las interacciones entre usuario y aplicación se utiliza un conjunto de casos de uso. La definición de la ERS se realiza dirigida por el cliente y el equipo de desarrollo. Por su importancia, es necesario el uso de un lenguaje y estructura adecuada, de forma que sea comprensible para cualquier parte involucrada en el proyecto. Para este TFG, se ha decidido realizar una distinción en:

- Requisitos funcionales.
- Requisitos de restricción
- Requisitos no funcionales.

Como método para una mejor comprensión de los requisitos, así como de su importancia, necesidad y prioridad, se ha decidido el desarrollo de la siguiente tabla. Se utiliza una tabla para la definición de cada requisito.

Identificador:	RS-XX
Nombre:	
Descripción:	
Fuente:	
Prioridad:	**Necesidad:**
Estabilidad:	**Verificabilidad:**
Prerrequisitos:	

Tabla 3-1: Ejemplo tabla de requisitos de software. (Fuente: Adaptación norma IEEE-830)

- **Identificador**: Para la identificación de cada requisito se le asignará un número de identificador único. Este número empezará por RS (requisito de software) seguido de un número.
- **Nombre:** Titular que resume el requisito.
- **Descripción:** Breve descripción sobre el objetivo del requisito.
- **Fuente:** Es el titular o persona que establece este requisito. En este TFG se considera fuente al usuario, para aquellos requisitos que cumplen las funciones del proyecto, y al desarrollador, para aquellos requisitos que implican la verificación del correcto funcionamiento del sistema.
- **Prioridad:** Nivel de importancia del requisito. Los posibles valores son alta, media o baja.
- **Necesidad:** Nivel de necesidad declarado por el usuario que establece el requisito. Los posibles valores son alta, media o baja.
- **Estabilidad:** Indicador de la posible variabilidad del requisito a lo largo del proyecto. Los posibles valores son alta, media o baja.
- **Verificabilidad:** Indicador de la facilidad de comprobación de un requisito. Los posibles valores son alta, media o baja.
- **Prerrequisito:** Posibles requisitos de los que depende el requisito definido.

3.2.1. Requisitos funcionales

Establecen los comportamientos del sistema. Un requisito funcional define una función del sistema o sus componentes. Pueden ser requisitos funcionales: los cálculos, los detalles técnicos, la manipulación de datos y otras funcionalidades específicas que el sistema deba cumplir.

Identificador:	RS-01
Nombre:	Calibrado sensores.
Descripción:	El sistema proveerá de una opción para el calibrado de sensores, y esta podrá ser utilizada en cualquier momento durante la ejecución del programa.
Fuente:	Usuario.
Prioridad: Alta.	**Necesidad:** Alta.
Estabilidad: Alta.	**Verificabilidad:** Alta.
Prerrequisitos:	No procede.

Tabla 3-2: RS-01: Calibrado sensores.

Identificador:	RS-02
Nombre:	Verificación calibrado.
Descripción:	El sistema proveerá de un medio para informar al usuario si la calibración ha terminado con éxito.
Fuente:	Usuario.
Prioridad: Alta.	**Necesidad:** Alta.
Estabilidad: Alta.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-01.

Tabla 3-3: RS-02: Verificación calibrado.

Identificador:	RS-03
Nombre:	Valores calibrado.
Descripción:	El sistema proveerá de un medio para mostrar los valores del sensor del acelerómetro, en tiempo real. Se mostrarán los valores para los ejes X, Y y Z, únicamente en caso de que el calibrado haya sido ejecutado con éxito.
Fuente:	Desarrollador.
Prioridad: Media.	**Necesidad:** Alta.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-01 y RS-02.

Tabla 3-4: RS-03: Valores calibrado.

Identificador:	RS-04
Nombre:	Detector de congestión en el tráfico.
Descripción:	El sistema detectará mediante un algoritmo, el estado de las carreteras, siempre que en estas haya algún tipo de congestión en el tráfico
Fuente:	Usuario.
Prioridad: Media.	**Necesidad:** Alta.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-01 y RS-02.

Tabla 3-5: RS-04: Detector de congestión en el tráfico.

Identificador:	RS-05
Nombre:	Mensaje evento de congestión.
Descripción:	El sistema mostrará al usuario un mensaje, indicando la detección de congestión vehicular en la vía por la que circula en ese instante, cuando un evento de tales características sea detectado por el propio sistema.
Fuente:	Usuario y desarrollador.
Prioridad: Alta.	**Necesidad:** Alta.
Estabilidad: Alta.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-01 y RS-04.

Tabla 3-6: RS-05: Mensaje evento de congestión.

Identificador:	RS-06
Nombre:	Informar manualmente congestión vehicular.
Descripción:	El sistema proveerá de un método, que permitirá al usuario informar de forma manual del estado de la carretera. Podrá indicar que existe congestión vehicular en el instante que él lo decida.
Fuente:	Usuario y desarrollador.
Prioridad: Media.	**Necesidad:** Media.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	No procede.

Tabla 3-7: RS-06: Informar manualmente congestión vehicular.

Identificador:	RS-07
Nombre:	Log de eventos de tráfico.
Descripción:	El sistema llevará de forma interna, un log donde registrará todos los eventos detectados. Por lo que guardará los eventos manuales, y automáticos de congestión vehicular.
Fuente:	Desarrollador.
Prioridad: Media.	**Necesidad:** Alta.
Estabilidad: Alta.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-01, RS-04 y RS-06.

Tabla 3-8: RS-07: Log de eventos de tráfico.

Identificador:	RS-08
Nombre:	Formato del log eventos de tráfico
Descripción:	El sistema deberá utilizar para el registro de eventos en un log, un formato que permita migrar los datos a sistemas de tratamiento de datos, de forma fácil y sencilla.
Fuente:	Desarrollador.
Prioridad: Media.	**Necesidad:** Media.
Estabilidad: Alta.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-07.

Tabla 3-9: RS-08: Formato del log eventos de tráfico.

3.2.2. Requisitos de restricción

Los requisitos de restricción son aquellos que deben cumplirse. De no hacerlo, el comportamiento de la aplicación se podría volver inestable e incorrecto.

Identificador:	RS-09
Nombre:	Permisos utilización recursos.
Descripción:	El sistema deberá de requerir al usuario, los permisos necesarios para el acceso a los recursos del sistema. De no ser concedidos, se deberá impedir el uso de la aplicación.
Fuente:	Desarrollador.
Prioridad: Alta.	**Necesidad:** Media.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-09.

Tabla 3-10: RS-09: Permisos utilización recursos.

Identificador:	RS-10
Nombre:	Sensores GPS y acelerómetro.
Descripción:	El sistema necesita del uso de los sensores de GPS y acelerómetro, por lo que el dispositivo Hardware utilizado deberá disponer de ellos.
Fuente:	Desarrollador.
Prioridad: Alta.	**Necesidad:** Alta.
Estabilidad: Alta.	**Verificabilidad:** Alta.
Prerrequisitos:	No procede.

Tabla 3-11: RS-10: Sensores GPS y acelerómetro.

Identificador:	RS-11
Nombre:	Conexión internet móvil.
Descripción:	El sistema deberá estar provisto de conexión a internet móvil, que posibilite la utilización de bibliotecas externas y el envío de datos.
Fuente:	Desarrollador.
Prioridad: Alta.	**Necesidad:** Alta.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	No procede.

Tabla 3-12: RS-11: Conexión internet móvil.

Identificador:	RS-12
Nombre:	Cobertura móvil y GPS.
Descripción:	El sistema deberá ser utilizado al aire libre, en zonas donde la cobertura móvil y GPS sea lo suficientemente fuerte.
Fuente:	Desarrollador.
Prioridad: Media.	**Necesidad:** Media.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-10 y RS-11.

Tabla 3-13: RS-12: Cobertura móvil y GPS.

Identificador:	RS-13
Nombre:	Memoria del dispositivo.
Descripción:	El dispositivo deberá contar con la memoria necesaria para el almacenamiento de datos.
Fuente:	Desarrollador.
Prioridad: Media.	**Necesidad:** Media.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	RS-10 y RS-11.

Tabla 3-14: RS-13: Memoria del dispositivo.

3.2.3. Requisitos no funcionales

También denominados atributos de calidad, es un requisito que especifica criterios o características de funcionamiento. Definen propiedades del sistema, como el tiempo de respuesta, necesidades de almacenamiento, fiabilidad, el uso de un tipo de herramienta, un lenguaje concreto o método de desarrollo.

Identificador:	RS-14
Nombre:	Guardar calibración.
Descripción:	Se habilitará una opción que permitirá guardar el calibrado actual.
Fuente:	Desarrollador.
Prioridad: Media.	**Necesidad:** Alta.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	No procede.

Tabla 3-15: RS-14: Guardar calibración.

Identificador:	RS-15
Nombre:	Cargar calibración.
Descripción:	Se habilitará una opción que permitirá cargar la última calibración guardada.
Fuente:	Desarrollador.
Prioridad: Media.	**Necesidad:** Alta.
Estabilidad: Media.	**Verificabilidad:** Alta.
Prerrequisitos:	No procede.

Tabla 3-16: RS-15: Cargar calibración.

Identificador:	RS-16
Nombre:	Privacidad.
Descripción:	La privacidad del usuario será fundamental. No se revelará ningún dato del mismo.
Fuente:	Usuario.
Prioridad: Alta.	**Necesidad:** Media.
Estabilidad: Alta.	**Verificabilidad:** Media.
Prerrequisitos:	No procede.

Tabla 3-17: RS-16: Privacidad.

Identificador:	RS-17
Nombre:	Algoritmo de detección.
Descripción:	El algoritmo desarrollado para la detección de congestión vehicular, se caracterizará por detectar este tipo de eventos de forma local en el dispositivo del usuario.
Fuente:	Desarrollador.
Prioridad: Alta.	**Necesidad:** Baja.
Estabilidad: Alta.	**Verificabilidad:** Media.
Prerrequisitos:	No procede.

Tabla 3-18: RS-17: Algoritmo de detección.

Identificador:	RS-18
Nombre:	Notificación de eventos.
Descripción:	La notificación de información a través de la red, se efectuará única y exclusivamente tras la detección de un evento de congestión vehicular. Además, se realizará con la menor información posible y necesaria.
Fuente:	Desarrollador.
Prioridad: Alta.	**Necesidad:** Baja.
Estabilidad: Alta.	**Verificabilidad:** Media.
Prerrequisitos:	RS-16 y RS-17.

Tabla 3-19: RS-18: Notificación de eventos.

Identificador:	RS-19
Nombre:	Desarrollo en Android.
Descripción:	El desarrollo de la aplicación móvil, se efectuará bajo la plataforma Android. Se utilizará además el entorno de desarrollo Android Studio.
Fuente:	Usuario, y desarrollador.
Prioridad: Media.	**Necesidad:** Baja.
Estabilidad: Alta.	**Verificabilidad:** Alta.
Prerrequisitos:	No procede.

Tabla 3-20: RS-19: Desarrollo en Android.

3.3. Casos de uso

Es una descripción de los pasos o actividades a tener en cuenta para llevar a cabo cualquier tipo de proceso. En software, esto implica una secuencia de interacciones entre un sistema y sus actores, en respuesta a un evento iniciado. Para el establecimiento de los casos de uso, se puede realizar un diagrama que especifica las comunicaciones y el comportamiento del sistema, mediante su interacción con los usuarios, y otros sistemas. [42]

Partiendo de los requisitos de usuario definidos en el apartado anterior, se procederá a describir los casos de uso aplicables a este proyecto. Cada caso de uso podrá satisfacer uno o varios de los requisitos.

Como método para una mejor comprensión de los casos de uso, y de forma adicional al desarrollo de diagramas, se ha decidido el desarrollo de la siguiente tabla. Se utiliza una tabla para la descripción de cada caso de uso:

Identificador:	CU-XX
Nombre:	
Actor/es:	
Descripción:	
Precondiciones:	
Postcondiciones:	
Secuencia principal	

Tabla 3-21: Ejemplo tabla casos de uso.

- **Identificador**: Para la identificación de cada caso de uso se le asignará un número de identificador único. Este número empezará por CU (caso de uso) seguido de un número.
- **Nombre:** Titular que resume el caso de uso.

- **Actor/es:** Representa el rol que toman los usuarios.
- **Descripción:** Breve descripción sobre el objetivo del caso de uso.
- **Precondiciones:** Especifica las condiciones que se deben cumplir previamente, para el cumplimiento del caso de uso actual.
- **Postcondiciones:** Especifica el estado en que queda el sistema tras el cumplimiento del caso de uso.
- **Secuencia principal:** Representa a la secuencia de pasos que componen principalmente al caso de uso. Estos pasos irán numerados en orden de ejecución.

A continuación, el diagrama de usos general de la aplicación separado por actores:

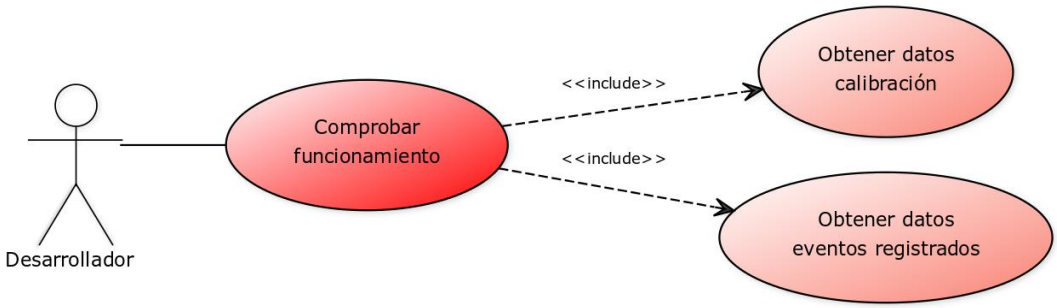

Figura 3.2: Diagrama casos de uso centrado en el actor Desarrollador

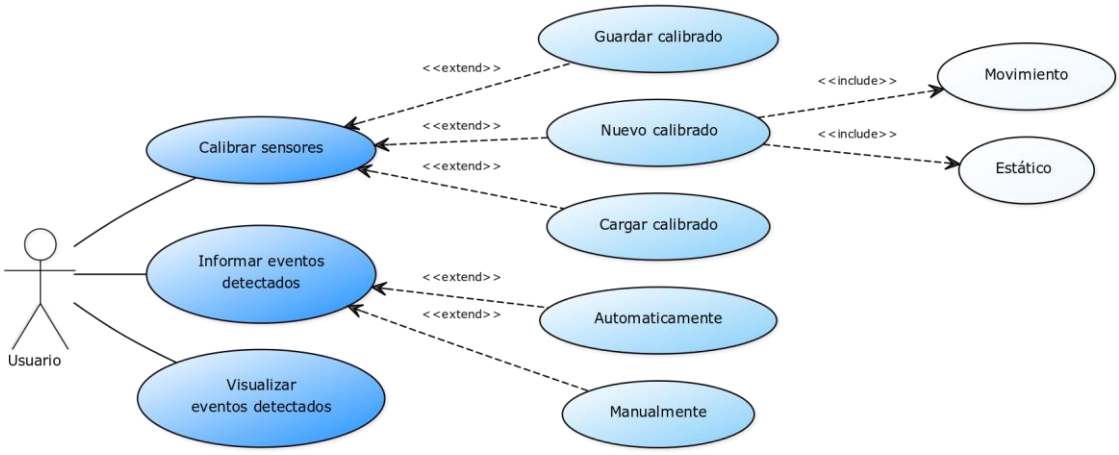

Figura 3.3: Diagrama casos de uso centrado en el actor Usuario.

Las figuras 3.2 y 3.3 desvelan que hay 4 posibles casos de uso generales a partir de los cuales derivan otros secundarios. Además, aparecen dos actores bien distinguidos:

- **Usuario:** Es aquel actor hacia el que va dirigido el proyecto. Es el usuario general, quien hará uso de la aplicación. Más adelante se explicarán los diferentes casos de uso que afectan a este actor.

- **Desarrollador:** Es el actor que se encarga de desarrollar y probar la aplicación. Sus casos de uso están definidos de cara al desarrollo y prueba del software (mantenimiento).

3.3.1. Casos de uso referentes al usuario

A continuación, pasamos a definir los casos de uso del diagrama anterior que afectan al actor usuario. Para ello usaremos la tabla indicada en este capítulo.

Identificador:	CU-01
Nombre:	Calibrar sensores.
Actor/es:	Usuario.
Descripción:	El usuario realiza alguna acción relacionada con el calibrado de sensores.
Precondiciones:	Ejecutar la aplicación.
Postcondiciones:	Se deberá seleccionar el tipo de calibración deseada.
Secuencia principal	1. Ejecutar aplicación. 2. Seleccionar acción de calibrado.
Casos de uso que extienden del actual:	Guardar calibrado (CU-02). Nuevo calibrado (CU-03). Cargar calibrado (CU-04).

Tabla 3-22: CU-01: Calibrar sensores.

Identificador:	CU-02
Nombre:	Guardar calibrado
Actor/es:	Usuario.
Descripción:	El usuario utiliza la acción guardar calibrado, que le permitirá guardar el calibrado actual.
Precondiciones:	Tener los sensores calibrados
Postcondiciones:	Los datos quedarán guardados para futuros usos.
Secuencia principal	1. Calibrado de sensores. 2. Cargar o realizar nuevo calibrado. 3. Guardar calibrado.

Tabla 3-23: CU-02: Guardar calibrado.

Identificador:	CU-03
Nombre:	Cargar calibrado.
Actor/es:	Usuario.
Descripción:	El usuario realiza la acción de cargar calibrado, que le permitirá cargar los parámetros guardados anteriormente.
Precondiciones:	Tener datos guardados.
Postcondiciones:	El sistema queda calibrado.
Secuencia principal	Se debe tener en cuenta que en otra secuencia anterior se habrá guardado datos de la aplicación. Estos datos podrán ser de la ejecución actual o de otras ejecuciones.

Tabla 3-24: CU-03: Cargar calibrado.

Identificador:	CU-04
Nombre:	Nuevo calibrado
Actor/es:	Usuario.
Descripción:	El usuario decide realizar una nueva calibración para los sensores.
Precondiciones:	Ejecutar la aplicación.
Postcondiciones:	El sistema quedará calibrado.
Secuencia principal	1. Calibrado de sensores 2. Nuevo calibrado
Casos de uso que incluye el actual:	Estático (CU-05). Movimiento (CU-06).

Tabla 3-25: CU-04: Nuevo calibrado.

Identificador:	CU-05
Nombre:	Estático
Actor/es:	Usuario.
Descripción:	El usuario permanece estático mientras los sensores toman datos para la calibración. El proceso durará unos 5 segundos.
Precondiciones:	Iniciado nueva calibración.
Postcondiciones:	El sistema pasará al calibrado en movimiento.
Secuencia principal	1. Calibrado de sensores. 2. Nuevo calibrado. 3. Estático.

Tabla 3-26: CU-05: Estático.

Identificador:	CU-06
Nombre:	Movimiento.
Actor/es:	Usuario.
Descripción:	El usuario inicia el movimiento del vehículo. El sistema tomará medidas para la calibración, detectando eventos de frenado.
Precondiciones:	Iniciado nueva calibración, calibración en estático terminada.
Postcondiciones:	El sistema quedará calibrado.
Secuencia principal	1. Calibrado de sensores. 2. Nuevo calibrado. 3. Estático. 4. Movimiento.

Tabla 3-27: CU-06: Movimiento.

Identificador:	CU-07
Nombre:	Informar eventos detectados
Actor/es:	Usuario.
Descripción:	El usuario realiza alguna acción relacionada con la información de eventos de congestión detectados.
Precondiciones:	Calibrar aplicación.
Postcondiciones:	El evento será comunicado y quedará registrado.
Secuencia principal	1. Ejecutar aplicación. 2. Calibrar aplicación. 3. Seleccionar acción de informar eventos.
Casos de uso que extienden del actual:	Automáticamente (CU-08). Manualmente (CU-09).

Tabla 3-28: CU-07: Informar eventos detectados.

Identificador:	CU-08
Nombre:	Automáticamente
Actor/es:	Usuario.
Descripción:	La detección de eventos de congestión de tráfico se realiza de forma automática
Precondiciones:	Informar de eventos
Postcondiciones:	Los datos quedarán guardados para futuros usos.
Secuencia principal	1. Seleccionar acción de informar eventos detectados. 2. Automática.

Tabla 3-29: CU-08: Automáticamente.

Identificador:	CU-09
Nombre:	Manualmente
Actor/es:	Usuario.
Descripción:	La detección de eventos de congestión de tráfico se realiza de forma manual.
Precondiciones:	Informar de eventos
Postcondiciones:	Los datos serán guardados para futuros usos.
Secuencia principal	1. Seleccionar acción de informar eventos detectados. 2. Manual.

Tabla 3-30: CU-09: Manualmente.

Identificador:	CU-10
Nombre:	Visualizar eventos detectados.
Actor/es:	Usuario.
Descripción:	El usuario desea visualizar los eventos detectados con anterioridad.
Precondiciones:	Eventos detectados.
Postcondiciones:	No procede.
Secuencia principal	1. Ejecutar aplicación. 2. Calibrar aplicación. 3. Detectar evento. 4. Visualizar evento.

Tabla 3-31: CU-10: Visualizar eventos detectados.

3.3.2. Casos de uso referentes al desarrollador

A continuación, pasamos a definir los casos de uso del diagrama anterior que afectan al actor desarrollador. Para ello usaremos la tabla indicada en este capítulo.

Identificador:	CU-11
Nombre:	Comprobar funcionamiento.
Actor/es:	Desarrollador.
Descripción:	El desarrollador desea comprobar el correcto funcionamiento de la aplicación.
Precondiciones:	Ejecutar aplicación.
Postcondiciones:	Se habrá detectado el correcto funcionamiento.
Secuencia principal	1. Ejecutar aplicación. 2. Calibrar aplicación. 3. Detectar evento. 4. Comprobar funcionamiento.
Casos de uso que incluye actual:	Obtener datos calibración (CU-12). Obtener datos eventos registrados (CU-13).

Tabla 3-32: CU-11: Comprobar funcionamiento.

Identificador:	CU-12
Nombre:	Obtener datos calibración.
Actor/es:	Desarrollador.
Descripción:	El desarrollador selecciona la opción de obtener datos sobre la calibración.
Precondiciones:	Comprobar funcionamiento.
Postcondiciones:	El sistema mostrará los valores actuales para los ejes X, Y y Z de la calibración.
Secuencia principal	1. Calibrar aplicación. 2. Comprobar funcionamiento. 3. Obtener datos calibración.

Tabla 3-33: CU-12: Obtener datos calibración.

Identificador:	CU-13
Nombre:	Obtener datos eventos registrados.
Actor/es:	Desarrollador.
Descripción:	El desarrollador selecciona la opción para la obtención de los datos sobre eventos registrados.
Precondiciones:	Comprobar funcionamiento.
Postcondiciones:	El desarrollador obtendrá uno, o varios ficheros con los datos guardados en sesiones anteriores. O bien recibirá datos enviados por la red de forma automática.
Secuencia principal	1. Calibrar aplicación. 2. Detección de eventos 3. Comprobar funcionamiento. 4. Obtener datos eventos registrados.

Tabla 3-34: CU-13: Obtener datos eventos registrados.

4. Implementación del sistema

En este capítulo se muestra la implementación de la solución técnica. Para ello se utilizará todo lo descrito en capítulos anteriores, así como la experiencia técnica adquirida en el lenguaje Android. Se seguirá la arquitectura descrita, y se cumplirán los objetivos y requisitos fijados. Además, se describirá la metodología utilizada para el desarrollo del software.

4.1. Desarrollo ágil

Durante el proceso de creación de este proyecto, se ha utilizado una metodología ágil[41]. Desde la perspectiva del software, se define agilidad como un comportamiento persistente que se caracteriza por su flexibilidad para adaptarse a los cambios, esperados o inesperados, de forma rápida. Además, busca lo económico, una duración corta de tiempo, sin perder en calidad. Se utilizan la experiencia y el conocimiento como instrumentos que completan esta mejora.

Las metodologías de tipo ágil se caracterizan por tener un ciclo de vida iterativo. Algunas otras características de este tipo de metodología son:

- Procesos iterativos e incrementales.
- Mitigación del riesgo.
- Mejora continua.
- Calidad desde la primera iteración
- Se priorizan los requerimientos según su valor.
- Colaboración continua con el cliente
- Incorporación de cambios.

Existen diversos tipos de metodologías de desarrollo ágil, atendiendo a diferentes filosofías, la elegida para este desarrollo es Scrum.

Scrum[29]**:** Se trata de una metodología de gestión y control para productos, cuyo objetivo es restar complejidad a dicha gestión. Se trata de focalizar los esfuerzos en un desarrollo de software que cumpla con los requisitos. Se caracteriza por ser una metodología simple, lo que la convierte en fácilmente escalable para distintos tipos de proyecto. Entre sus características más destacables están:

- Adopción de una estrategia de desarrollo incremental.
- Basar la calidad del resultado en el conocimiento e interacción del equipo desarrollador.
- Solapamiento de las distintas fases de desarrollo.

Para entender el proceso iterativo de Scrum, es necesario comprender que su mayor utilidad es durante el desarrollo de proyectos de investigación y desarrollo. Cada iteración permitirá tener un producto de calidad, y además la introducción de características nuevas.

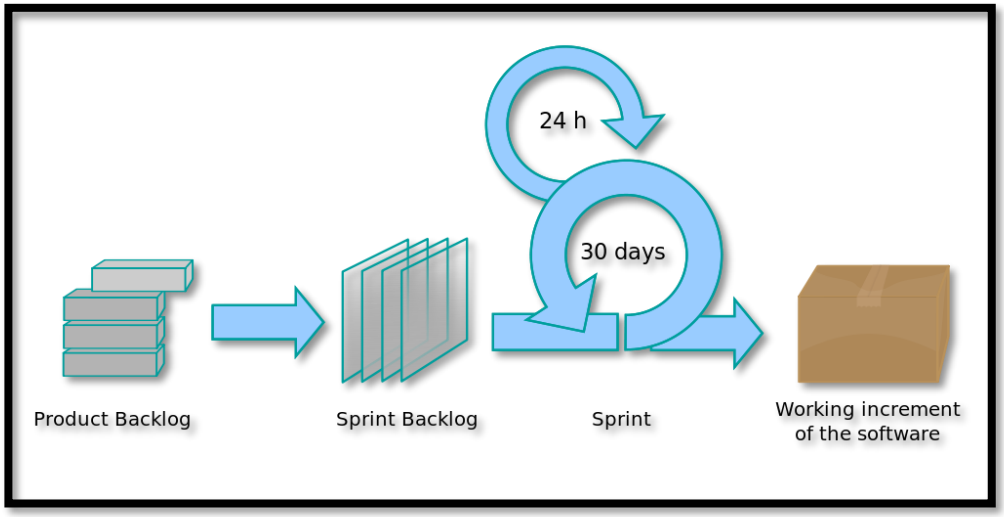

Figura 4.1: Diagrama de la metodología Scrum. (Fuente Wikipedia)

1. Al comenzar cada iteración, el equipo decidirá las características a implementar.
2. Con las características elegidas, se plantea la mejor forma de implementarlas.
3. Se realiza el desarrollo.
4. Una vez terminado el desarrollo, se hace una revisión de lo implementado, y se analiza la siguiente iteración.

A pesar de la baja complejidad de este proyecto, y del reducido número de actores, la decisión de este tipo de metodología se basa en:

- Ir consiguiendo objetivos a la vez que el desarrollador adquiere conocimientos.
- El tutor podrá comprobar el avance y los resultados tras cada iteración.
- Entre el desarrollador y el tutor se establecen nuevos objetivos a perseguir en las siguientes iteraciones.
- Se consigue desde el principio software de calidad que se podrá reutilizar tras cada iteración.

En base a lo explicado, se definen las siguientes secciones que dividen de forma bastante fiel, cada una de las iteraciones utilizadas durante el desarrollo del proyecto.

4.1.1. Aplicación GPS[1]

A pesar de contar con conocimientos de Java y XML, ha sido necesaria una primera aproximación a Android y su método de desarrollo, así como el funcionamiento de su IDE, Android Studio. Tras finalizar estos primeros pasos, se procede al desarrollo de distintas aplicaciones que facilitarán la familiarización con el funcionamiento de los sensores.

La primera de estas aplicaciones, se utiliza para aprender a utilizar el sensor GPS de los dispositivos móviles Android. Este sensor, como ya vimos en capítulos anteriores, permite la geo-localización. Utilizaremos los datos obtenidos del sensor GPS para ubicar aquellos puntos de la red viaria donde se registren eventos.

Para el manejo de datos de localización, Android cuenta con varias clases. La clase principal se denomina `LocationManager`. Esta clase provee acceso al servicio de localización del sistema, permitiendo a las aplicaciones obtener actualizaciones periódicas de la localización geográfica.

Como método para minimizar los recursos del sistema consumidos, esta clase cuenta con distintos dispositivos para proporcionar la localización, algunos de ellos son: GPS o red. El desarrollador podrá escoger en función de sus necesidades de precisión, consumo de batería, número de actualizaciones en el tiempo, etc.

Este proyecto requiere de una localización precisa, que permita ubicar perfectamente los eventos en un punto concreto de una vía concreta, por lo que se decide configurar `LocationManager` para obtener la ubicación mediante GPS. Esto consumirá más batería, pero lo gestionaremos encendiendo el GPS sólo cuando sea necesario.

En la siguiente figura mostramos la interfaz de la APP GPS desarrollada. Esta mostrará la ubicación en términos de latitud, longitud, y altitud. De forma adicional se informará de la precisión y la velocidad actual del dispositivo. Además, está provisto de botones de activar y desactivar. Todas estas características y datos serán de utilidad en el desarrollo de la APP final. [17]

[1] La APP descrita en esta sección, forma parte de las realizadas de forma común (Ver Nota al lector) [43], con el fin de facilitar el aprendizaje de ambos autores sobre Android y el uso de las bibliotecas para el GPS.

Figura 4.2: Interfaz de la aplicación GPS.

4.1.2. Aplicación acelerómetro[2]

Tras el desarrollo de una APP capaz de manejar el sensor GPS, y que servirá como módulo para la construcción de la APP final, se decide desarrollar una aplicación para el uso del acelerómetro en la siguiente iteración.

Al igual que ocurre con el sensor GPS, Android cuenta con varias clases para el manejo del acelerómetro y los eventos de este sensor. Durante la implementación de esta aplicación, es muy importante tener en cuenta el sistema de coordenadas definido.

El sistema de coordenadas definido por la API de Android [17] es el que podemos observar en la siguiente figura. Está definido por la posición de la pantalla. Los ejes no son intercambiables aunque haya cambios en la orientación del dispositivo.

[2] La APP descrita en esta sección, forma parte de las realizadas de forma común (Ver Nota al lector) [43], con el fin de facilitar el aprendizaje de ambos autores sobre Android y el uso de las bibliotecas para el acelerómetro.

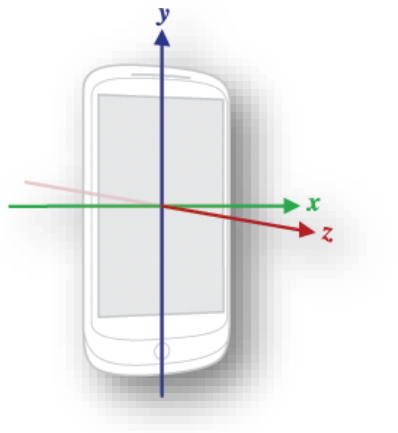

Figura 4.3: Sistema de coordenadas en Android.(Fuente Web desarrolladores Android)

El eje X es horizontal y apunta hacia la derecha. El eje Y es vertical y apunta hacia arriba. El eje Z es trasversal y apunta hacia el fondo de la pantalla.

Como veremos en sucesivas secciones, este TFG centra su algoritmo de detección, en la medición de la aceleración para la detección de eventos. Esto hace que el acelerómetro sea un dispositivo de gran importancia.

En la aplicación desarrollada se utiliza la API de Android para obtener la medida de este sensor en los tres ejes. Está permitido la configuración del tiempo entre eventos, y otros parámetros (aunque estos no nos serán útiles por estar más dirigidos a aplicaciones como juegos).

En la siguiente figura mostramos la interfaz de la APP acelerómetro desarrollada. Esta muestra los valores de aceleración para los distintos ejes, con un refresco del orden de milisegundos. Tan solo se necesita iniciar la aplicación para que comience a mostrar datos.

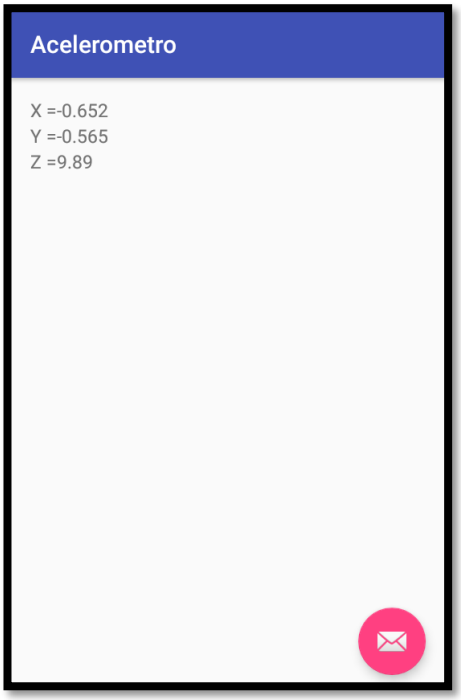

Figura 4.4: Interfaz de la aplicación acelerómetro.

4.1.3. Primera aproximación – Calibrado[3]

Recordamos que el objetivo de este TFG es encontrar una forma alternativa, para la detección de eventos de congestión en el tráfico de las vías. En la actualidad, tal y como vimos en el Capítulo 2 al estudiar el estado del arte, existen varios algoritmos de detección: el de Google se basa en la obtención continua de datos del usuario, que envía a sus servidores, lo cual no es ni eficaz, ni permite una gran privacidad del usuario. El de Waze es incómodo, ya que el usuario tendrá que avisar manualmente de los eventos. En ambos casos el análisis de datos, y la decisión de si existe congestión se realiza en servidores de terceros, ajenos al usuario.

La solución implementada en este TFG (basada en el artículo *TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones* [27]), trata de dar la vuelta a las soluciones actuales. El punto de vista es que contamos en la actualidad con micro-ordenadores al alcance de nuestra mano, con amplia capacidad de procesado, e infinidad de sensores. Estos son los Smartphone y otros dispositivos móviles.

Esa es la idea que persigue esta aplicación, mediante el uso de los sensores del Smartphone, se detectarán localmente eventos de tráfico y se

[3] La APP descrita en esta sección, forma parte de las realizadas de forma común (Ver Nota al lector) [43], con el fin de facilitar el desarrollo de una base común: La reorientación del acelerómetro.

decidirá si son o no, congestión viaria. De forma general el algoritmo se define en los siguientes pasos:

1. Detectar si existe una frenada.
2. Medir la velocidad del usuario en ese momento.
3. Consultar la velocidad máxima de la vía.
4. Decidir si existe evento de congestión.
5. Informar del evento si procede.

En esta iteración, el objetivo es crear una APP capaz de detectar las frenadas de un vehículo, y para ello se utilizará el sensor acelerómetro. Es importante tener en cuenta la orientación del dispositivo móvil dentro del vehículo. Si no calibramos virtualmente de forma correcta los ejes del dispositivo, para que coincidan con los ejes del vehículo, será muy difícil medir las frenadas y obtendremos grandes errores en la detección de frenada. Antes de comenzar a dar más detalles, es importante establecer una nomenclatura clara que explique los ejes de cada sistema. Utilizaremos las letras minúsculas x, y, z para representar los ejes del sensor/Smartphone, tal y como fue descrito en la figura 4.3. En el caso del vehículo se utilizarán las letras mayúsculas X, Y, Z para representar sus ejes, de tal forma que el X apunta directamente al frente, el Y a la derecha, y el Z hacia el suelo del vehículo.

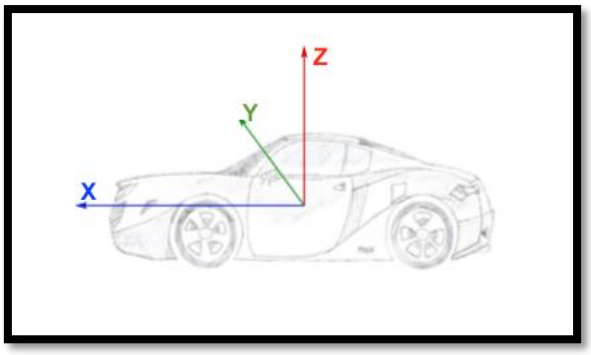

Figura 4.5: Sistema de coordenadas del vehículo.

Si x, y ,z están alineados con X, Y, Z, decimos que el acelerómetro está bien orientado. En cualquier otro caso el acelerómetro estará desorientado. Utilizaremos también la siguiente nomenclatura para nombrar las lecturas de los distintos ejes $\boldsymbol{a_x, a_y, a_z}$ y $\boldsymbol{a_X, a_Y, a_Z}$. Se debe tener en cuenta que incluso un acelerómetro bien orientado, marcará un valor de 9.8 m/s^2 en el eje Z debido a la atracción gravitatoria.

Determinando la orientación del acelerómetro:

De forma general, un dispositivo móvil (y por consiguiente su acelerómetro) y sus ejes x, y, z, pueden estar en cualquier posición con respecto al vehículo y sus ejes X, Y, Z. Además, esta orientación puede variar a lo largo del tiempo durante el trayecto. Un dispositivo desorientado, hace

difícil medir las condiciones del tráfico mediante las medidas del acelerómetro. De ahí la importancia de la reorientación virtual del sensor, para compensar su desorientación.

Así definimos la forma canónica x', y', z' que corresponde a X, Y, Z una vez el sensor está orientado. Cualquier orientación arbitraria de los ejes x, y, z puede convertirse en X, Y, Z aplicando sus ejes de rotación en una secuencia canónica. Nuestro objetivo será calcular los ángulos de rotación para cada eje.

Mediante el uso de los ángulos de Euler, podemos simplificar estos cálculos necesarios de forma significativa. Hay muchas formulaciones distintas para Euler, pero nosotros utilizaremos la denominada rotación Z-Y-Z. Cualquier orientación del acelerómetro podrá ser descrita por un ángulo de pre-rotación Φ_{pre} sobre Z, un ángulo de inclinación θ_{inc} sobre Y, y un ángulo de post-rotación ψ_{post} también sobre Z.

1. **Estimando los ángulos de pre-rotación e inclinación:** Con estas consideraciones, podemos describir un procedimiento para determinar la orientación de un acelerómetro desorientado. Si un acelerómetro se encuentra en estado de reposo, la única aceleración detectada será la causada por el efecto de la gravedad, en el eje Z. La inclinación viene dada por cambios en el eje z con respecto a Z, así que se debe cumplir que $\boldsymbol{a_z = a_Z \cos(\theta_{inc})}$. Sabiendo que $\boldsymbol{a_Z = 9.8\ m/s^2}$, se cumplirá que:

$$\boldsymbol{\theta_{inc} = cos^{-1}(a_z)}$$

Ecuación 4.1: Fórmula del ángulo inclinación.

Pre-rotación seguido de inclinación dará como resultado un valor distinto de cero en $\boldsymbol{a_x}$ y $\boldsymbol{a_y}$ debido al efecto de la gravedad. Lo que nos da como resultado (para mayor detalle ver fuente bibliográfica [27]):

$$\boldsymbol{\phi_{pre} = tan^{-1}(\frac{a_y}{a_z})}$$

Ecuación 4.2: Fórmula del ángulo de pre-rotación.

Para poder estimar Φ_{pre} y θ_{inc} es necesario identificar aquellos periodos donde el dispositivo móvil se encuentra en reposo. Capturaremos medidas para los ejes $\boldsymbol{a_x, a_y, a_z}$ durante aproximadamente 5 segundos. Realizaremos la mediana de estos valores, y podremos utilizar las fórmulas arriba indicadas para realizar el cálculo de los ángulos.

Por tanto, en esta parte de nuestra aplicación, el usuario pulsará un botón cuando quiera calibrar, y se tomarán medidas durante 5 segundos en reposo para el cálculo de Φ_{pre} y θ_{inc} con las fórmulas.

2. **Estimando el ángulo de post-rotación:** La post-rotación se produce sobre el eje Z, por lo que el efecto de la gravedad no tiene impacto sobre él. Debemos entonces buscar una fuerza que no sea paralela al eje Z, para poder estimar el valor del ángulo ψ_{post}. De forma práctica necesitaremos un campo de fuerza significativo en una dirección ortogonal al eje Z.

Podemos utilizar la fuerza que se produce cuando un coche frena hasta pararse, ya que esta será significativa en el eje X. Utilizando el sensor GPS, y los datos provistos por el mismo (como vimos en la sección 4.1.1 la velocidad es uno de ellos), se detecta un momento de frenado en el vehículo y se capturan los valores de $\boldsymbol{a_x, a_y, a_z}$ durante al menos 3 segundos después de que este ocurra. Con estos datos podremos proceder al cálculo de ψ_{post} (para mayor detalle ver fuente bibliográfica [27]):

$$\boldsymbol{\psi_{post} = arctan\left(\frac{-a_x \sin(\phi_{pre}) + a_y \cos(\phi_{pre})}{(a_x \cos(\phi_{pre}) + a_x \sin(\phi_{pre}))\cos(\theta_{inc}) - a_z \sin(\theta_{inc})}\right)}$$

Ecuación 4.3: Fórmula del ángulo de post-rotación.

Por tanto, en esta parte, nuestra aplicación activará el GPS y esperará detectar un momento de fuerte frenado. Tomará medidas y realizará el cálculo de ψ_{post}.

A partir de estos ángulos, y teniendo en cuenta que la formulación de Euler utilizada es Z-Y-Z, tenemos todo lo necesario para calcular nuestra matriz de rotación. Una matriz de rotación es una matriz que representa una rotación en el espacio euclídeo. Por definición, esta matriz de rotación cumplirá con ser ortogonal y de determinante con valor uno. El cálculo de la matriz de rotación está basado en las fórmulas del libro *Dynamics of Tree-Type Robotic Systems*[44].

$$\boldsymbol{R_\phi \begin{pmatrix} \cos(\phi_{pre}) & \sin(\phi_{pre}) & 0 \\ -\cos(\phi_{pre}) & \cos(\phi_{pre}) & 0 \\ 0 & 0 & 1 \end{pmatrix}}$$

Ecuación 4.4: Matriz para el ángulo de pre-rotación.

$$R_{\psi}\begin{pmatrix} \cos(\psi_{post}) & \sin(\psi_{post}) & 0 \\ -\operatorname{sen}(\psi_{post}) & \cos(\psi_{post}) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Ecuación 4.5: Matriz para el ángulo de post-rotación.

$$R_{\psi}\begin{pmatrix} \cos(\theta_{inc}) & 0 & -\sin(\theta_{inc}) \\ 0 & 1 & 0 \\ \sin(\theta_{inc}) & 0 & \cos(\theta_{inc}) \end{pmatrix}$$

Ecuación 4.6: Matriz para el ángulo de inclinación.

Una vez calculadas las matrices de cada uno de los ángulos, podemos calcular la matriz de rotación final. Esta matriz es de la forma:

$$R = R_{\Psi} \cdot R_{\theta} \cdot R_{\phi}$$

Ecuación 4.7: Cálculo de la matriz de rotación.

Si se realizan bien los diferentes pasos, esta matriz nos permitirá encontrar la correspondencia para los valores de a_X, a_Y, a_Z, a partir de los valores medidos en a_x, a_y, a_z, para un sensor desorientado. Tan solo será necesario realizar el siguiente cálculo (suponiendo a_X, a_Y, a_Z, iguales a $a_{x'}$, $a_{y'}$, $a_{z'}$, siendo estos últimos los valores del sensor reorientado): [27]

$$\begin{pmatrix} a_X \\ a_Y \\ a_Z \end{pmatrix} = R \cdot \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$

Ecuación 4.8: Cálculo de los valores para el sensor reorientado.

En la siguiente figura, mostramos la interfaz de la APP calibrado del acelerómetro. Esta mostrará los valores de aceleración para los distintos ejes en el sensor desorientado, con un refresco del orden de milisegundos. El usuario deberá pulsar el botón calibrar y seguir las instrucciones para reorientar el acelerómetro. Una vez reorientado aparecerán los valores de X, Y, Z. En el ejemplo se ve como el sensor ha sido reorientado, detectando la gravedad en el eje Z del coche.

Figura 4.6: Interfaz de la aplicación acelerómetro calibrado.

4.1.4. Segunda aproximación – Toma de datos

Una vez probada la APP que calibra el sensor acelerómetro, reorientándolo, y tras ver su correcto funcionamiento se pasa a la siguiente iteración. Sabemos, por recomendación del artículo, que el umbral de detección para frenadas se sitúa en 2.6m/s^2 (para frenadas más o menos agudas, que supongan una disminución de velocidad notoria). Esto implicaría una detección en nuestro acelerómetro calibrado de $\boldsymbol{a_X}$= -2.6 m/s^2.

Ahora, es necesario establecer algún tipo de patrón, que de detectarse suponga la posibilidad de un evento de congestión en el tráfico. Para ello, algo de gran utilidad es el estudio de los datos extraídos de los sensores de nuestro dispositivo móvil. Por tanto, se modificará la aplicación de calibrado del sensor, para que sea capaz de tomar muestras en sus datos, que puedan a posteriori ser analizadas por el desarrollador.

Se decide utilizar tres fuentes para la obtención de información, que podrá ser analizada en el momento o a posteriori:

1. La primera es la instalación en la aplicación de dos mensajes de texto. Estos mensajes se activarán al detectarse una aceleración o un frenado. En verde, se mostrará un mensaje con el texto ‘Acelerando’ (al detectarse un evento de aceleración), en amarillo auto, se mostrará un mensaje con el texto ‘Frenando’ (al detectarse un evento de frenado).
2. La segunda es utilizar el sistema GPS, para la medida continua de la velocidad en el vehículo. Estas medidas podrán ser analizadas a

posteriori para detectar patrones de frenado, o disminución fuerte de la velocidad que puedan indicar algún tipo de evento en el tráfico.

3. La tercera, es la introducción de un mecanismo para que el desarrollador en sus pruebas, pueda informar de forma manual de la existencia de un evento de congestión. Este mecanismo será un botón con el nombre ´Tráfico'.

Además de estas nuevas características, por basarse en la iteración anterior, la aplicación mantendrá sus opciones de calibrado, así como su muestra de los valores calibrados en los ejes X, Y, Z.

En la siguiente figura mostramos la interfaz de la APP para la toma de datos. Podemos observar todos los componentes anteriormente indicados.

Figura 4.7: Interfaz de la aplicación toma de datos.

Los resultados de los datos para comprobar si la calibración es correcta, se visualizan directamente en la interfaz de la aplicación. Para el análisis de los datos de velocidad, y de eventos generados manualmente, se utilizan ficheros, donde estos serán guardados. Posteriormente se utiliza el programa Matlab para la representación de los mismos. A continuación, mostramos una figura que combina ambos tipos de datos generada con este programa.

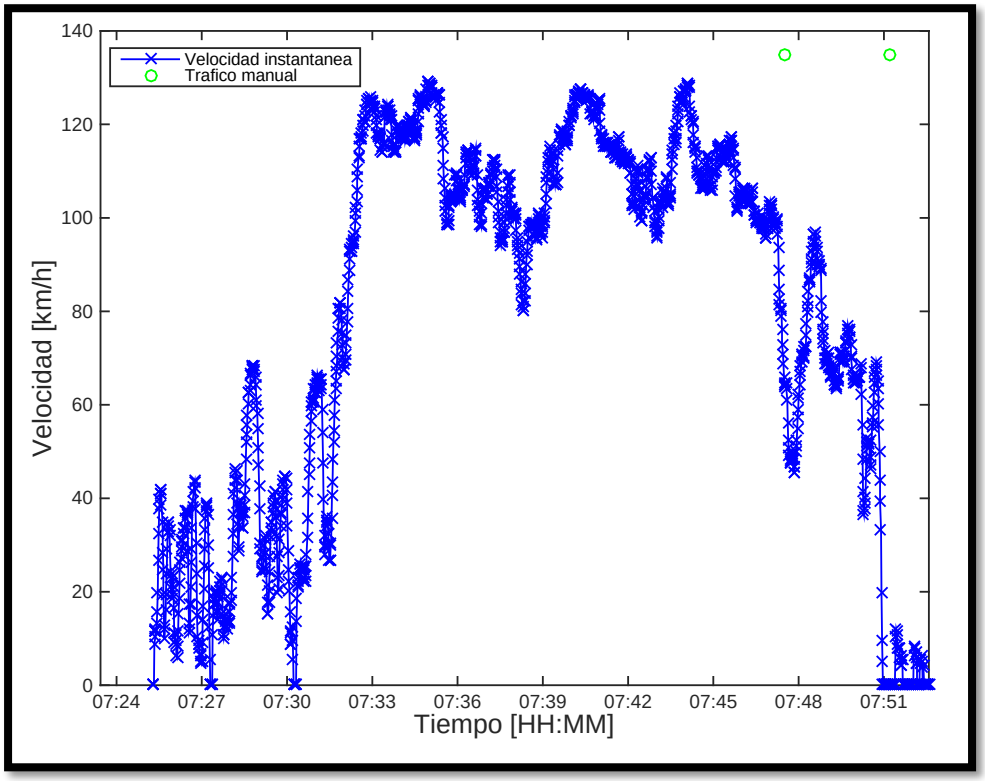

Figura 4.8: Gráfica de la velocidad instantánea del vehículo y los eventos manuales de tráfico.

Se debe tener en cuenta que a partir de ahora para el análisis de datos, se utiliza siempre un recorrido similar, en una misma franja horaria. Si analizamos la gráfica de la velocidad instantánea del vehículo a lo largo del recorrido (azul), resulta complicado visualizar cuando una frenada se produce por un evento de congestión, y cuando se realiza por cualquier otro tipo de evento. Sin embargo, al añadir la información proporcionada por el desarrollador durante el recorrido, que ha pulsado de forma manual dos eventos de tráfico, sí que podemos visualizar un cambio brusco en la velocidad.

La conclusión es clara, estos datos son insuficientes para la detección de eventos de congestión. Es necesario buscar otra alternativa, que bien por sí sola, o bien sumada a los datos ya obtenidos, pueda distinguir los eventos correctamente.

Por otro lado durante las pruebas de la APP, se verifica el correcto funcionamiento de la detección de frenados. Por lo que se da por preciso el umbral establecido (Como muestra, en la figura 4.7, se observa el rótulo ‘Frenando’ durante una frenada al testear la aplicación).

4.1.5. Tercera aproximación – Comprobación

Durante esta iteración, el esfuerzo se centra en buscar una forma de combinar los datos, que nos permita crear un algoritmo que detecte eventos de congestión en la vía. Para esta tarea, será útil el estudio realizado sobre las librerías de terceros, fáciles de implementar en Android.

De esta forma, tras valorar los aspectos ofrecidos, se elige Here WeGo como pieza clave de nuestro algoritmo. Here WeGo ofrece mediante protocolo HTTP, diferente información sobre tráfico. Para ello tan sólo se deberá construir la consulta correcta, en la que se incluye latitud y longitud. Por ejemplo http://route.st.nlp.nokia.com/routing/7.2/getlinkinfo.json?waypoint=40.344933,-3.723577&app_id=xxxxxxxxxxxxxxx&app_code=xxxxxxxxxxxxxxxxxxs
La respuesta recibida será con formato JSON, similar a la de la siguiente figura.


```
{"response":{"metaInfo":
{"mapVersion":"8.30.63.156","moduleVersion":"7.2.72.0-
55447","interfaceVersion":"2.6.25","timestamp":"2016-09-
06T12:08:38Z"},"link":[{"linkId":"-549330750","shape":
["40.3451228,-3.7235498","40.3449082,-3.723582"],"speedLimit":
25.0}]}}
```


Figura 4.9: Ejemplo de consulta API Here WeGo.

Lo particular de esta información, es que entre ella encontramos la denominada “speedLimit”. “speedLimit” es la velocidad máxima de la vía a la que está permitido la circulación (su valor viene expresado en m/s). La API de Here WeGo proporciona para cada tramo de la red viaria la velocidad máxima establecida.

Partiendo de la aplicación de toma de datos, y utilizando la velocidad instantánea y la velocidad máxima de la vía, podremos detectar de forma local si el usuario va por debajo de una velocidad, denominada de ‘confort’. Estableceremos el límite en 0.71 veces la máxima velocidad de la vía. Por ejemplo, si un vehículo en una vía con velocidad máxima de 120Km/h, circula por debajo de 85,2Km/h, podremos considerar que existe un evento de congestión.

Partiendo de estas premisas, se realizan las modificaciones pertinentes, añadiendo a lo ya medido, el almacenamiento de los datos de velocidad máxima, para los diferentes tramos de la vía durante la navegación. Posteriormente estos datos podrán ser analizados como hicimos anteriormente.

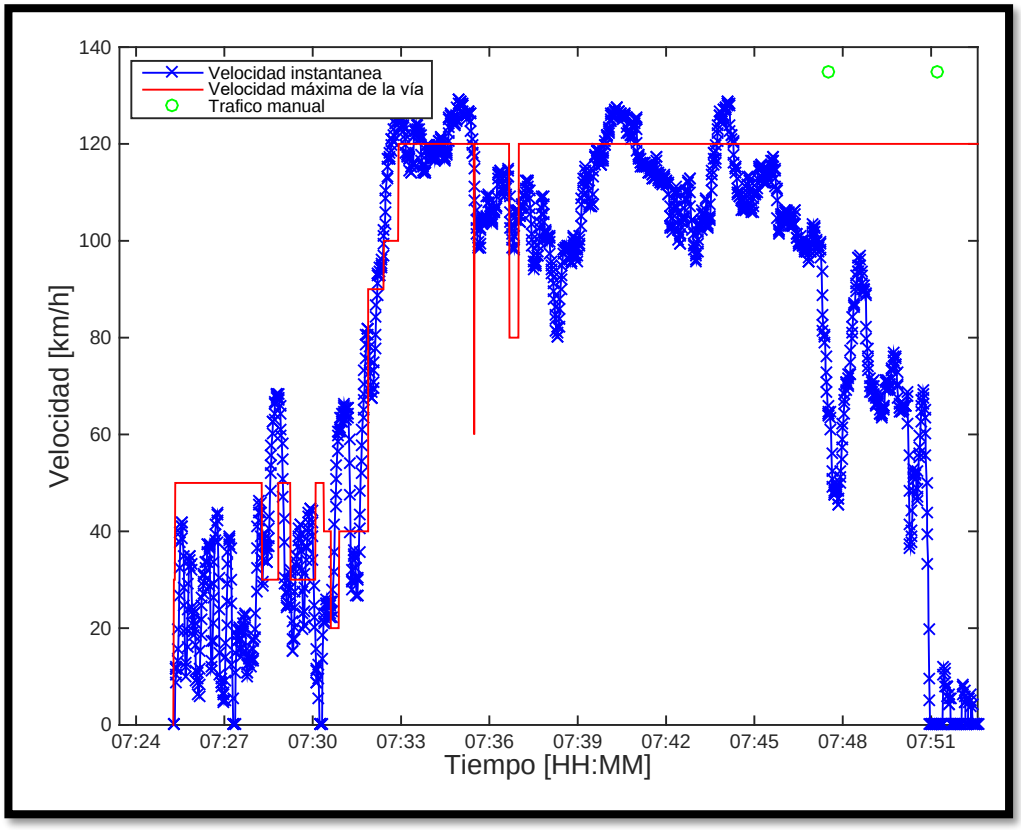

Figura 4.10: Gráfica de la velocidad instantánea del vehículo, los eventos manuales de tráfico y la velocidad máxima de la vía.

En cuanto a la interfaz de usuario, en esta versión de la aplicación no varía mucho con respecto a su última iteración.

4.1.6. Aplicación final

Con lo aprendido y desarrollado en las diferentes iteraciones, estamos preparados para el desarrollo de la aplicación final. Las conclusiones de la última iteración, nos empujan a establecer el funcionamiento final de la APP. Para ello se utilizará nuevamente como base la aplicación de esta última iteración. Se modificará añadiendo la capacidad de detección automática de eventos de congestión.

Para el diseño de esta capacidad de detección automática, será necesario el uso del detector de frenadas 'bruscas' desarrollado anteriormente. De esta manera al detectar una frenada, el sistema medirá la velocidad actual del usuario durante los siguientes 10 segundos. Realizará la media, y hará lo mismo con la velocidad máxima de la vía. Una vez realizados estos cálculos volveremos a utilizar la llamada velocidad de 'confort' (como ya sabemos, calculada sobre la velocidad máxima), para definir si existe un evento de congestión en la vía, o no. Por último, de existir, este será notificado al usuario, y a otros posibles usuarios de la aplicación.

Durante la ejecución, guardaremos los datos de detección automática. Esto nos permitirá comparar de forma gráfica, los datos de congestión detectados automáticamente, con los datos provistos de forma manual por el usuario. A continuación, mostramos una gráfica sobre ello.

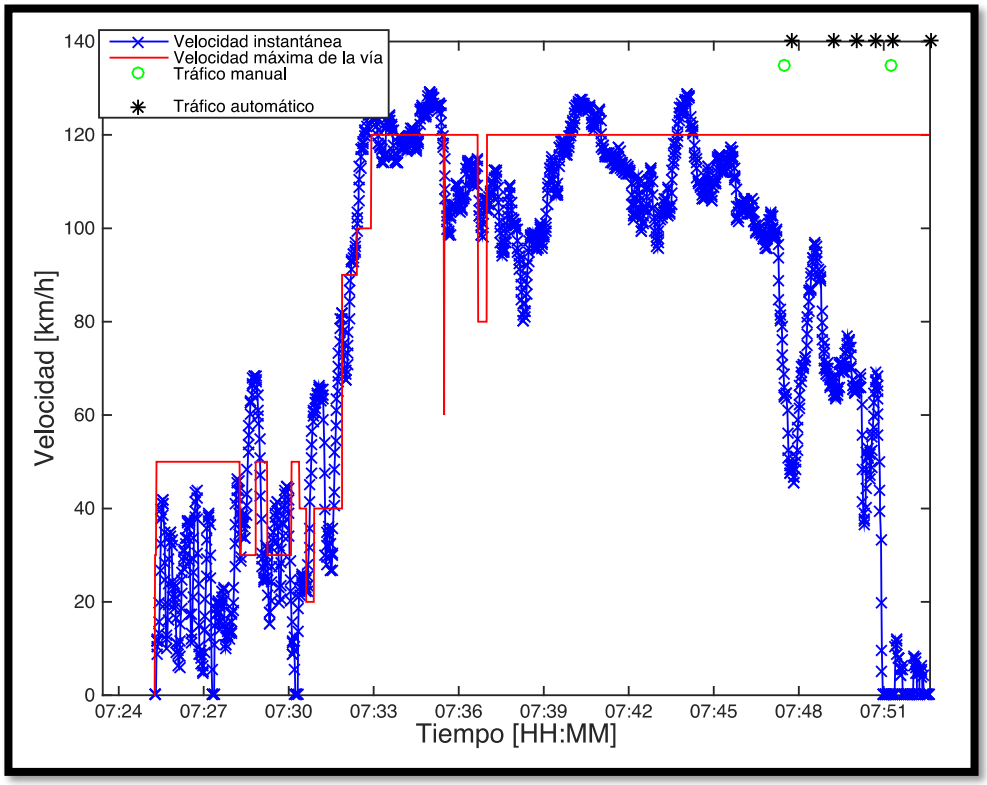

Figura 4.11: Gráfica de la velocidad instantánea del vehículo, la velocidad máxima de la vía, los eventos manuales de tráfico y los eventos automáticos.

En cuanto a la interfaz gráfica, su apariencia se basa en iteraciones anteriores, por lo que su estructura general será notablemente similar. Se introducen algunos cambios para que el aspecto sea más visual, y genere mayor agrado al usuario. Se ha introducido un mapa donde con un punto azul, se indica la ubicación en tiempo real del usuario. Debemos tener en cuenta que se añade información adicional, esta sirve para notificar al usuario un evento de congestión detectado de forma automática. Se notificará al usuario el evento en forma de círculo rojo y blanco. Además, si se produce un evento manual de tráfico, habiendo el usuario pulsado el botón, se añadirá un circulo de color azul y blanco similar al utilizado en la detección de tráfico automática.

De forma adicional se añadirá un menú desplegable, que permitirá guardar y cargar una configuración de calibrado. Esto facilitará la toma de datos, o la simple conducción, al evitar el calibrado del dispositivo cada vez que se inicia la APP.

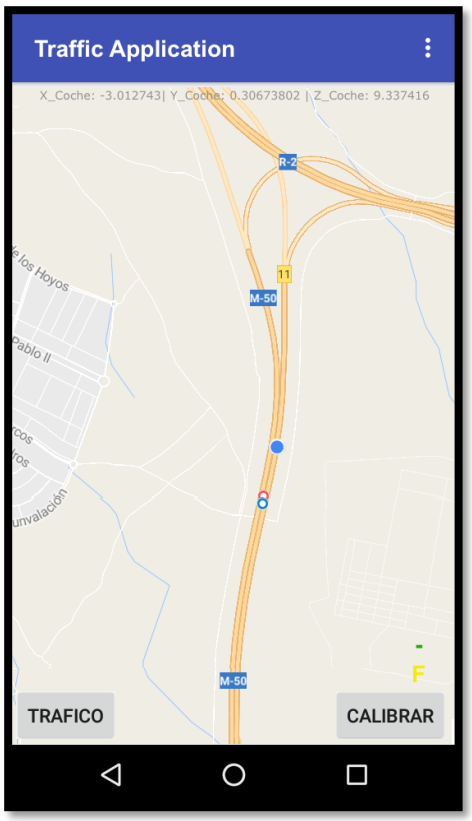

Figura 4.12: Interfaz de la aplicación final.

Figura 4.13: Permisos de usuario.

Por último, se añade a la aplicación la obligación de aceptar los permisos de usuario. Es decir, que el usuario será informado de la necesidad de conocer su ubicación, y se le requerirá permiso para acceder a la misma.

4.2. Diagrama de flujo

También denominado diagrama de actividades, sirve como representación gráfica de un algoritmo o proceso. A continuación, se mostrará un diagrama de flujo general para la aplicación. Este pretende ilustrar las acciones principales del sistema. Además, se adjuntarán dos diagramas de flujo adicionales, que cubren las actividades más importantes, de forma detallada:

- Fase de calibración (orientado del acelerómetro).
- Fase de detección de eventos (detección eventos de congestión).

Es importante señalar, que estos diagramas de flujo son tan sólo una aproximación, ya que un programa de estas características puede no tener un flujo lineal debido a ser manejado por el usuario. Es decir, el usuario puede realizar las acciones en el orden inverso, o cerrar la aplicación cuando él decida.

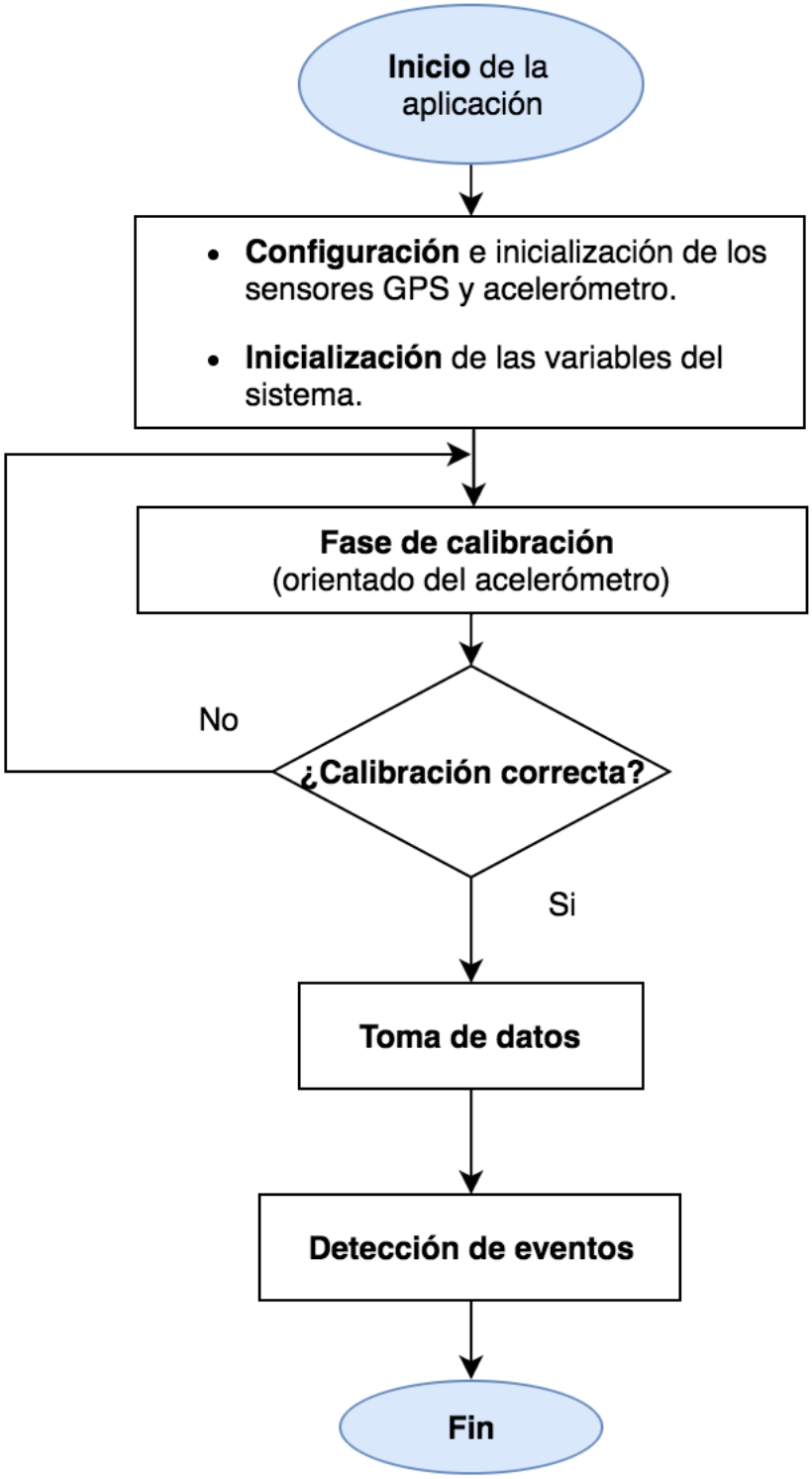

Figura 4.14: Diagrama de flujo general para la aplicación.

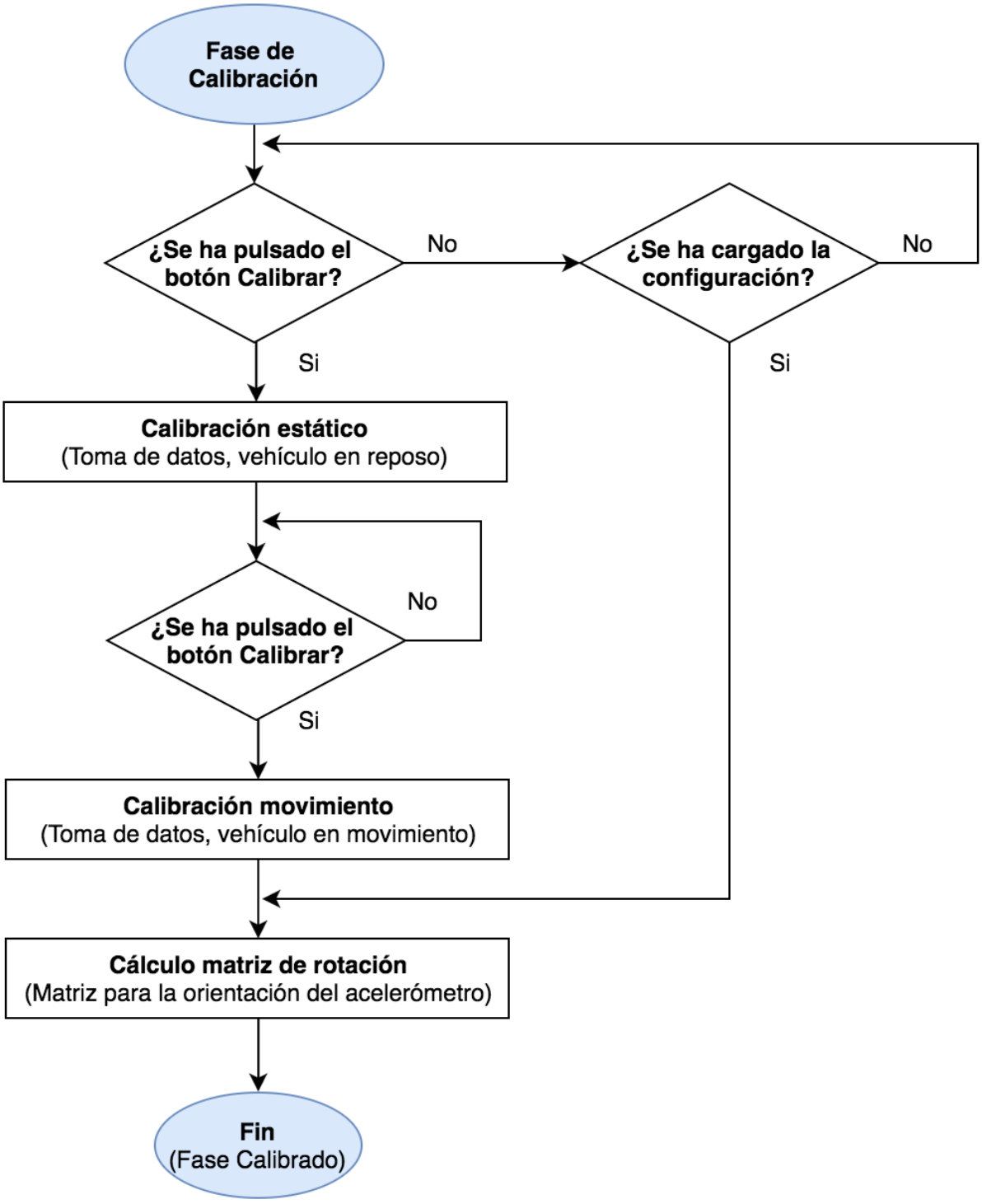

Figura 4.15: Diagrama para la fase de Calibración.

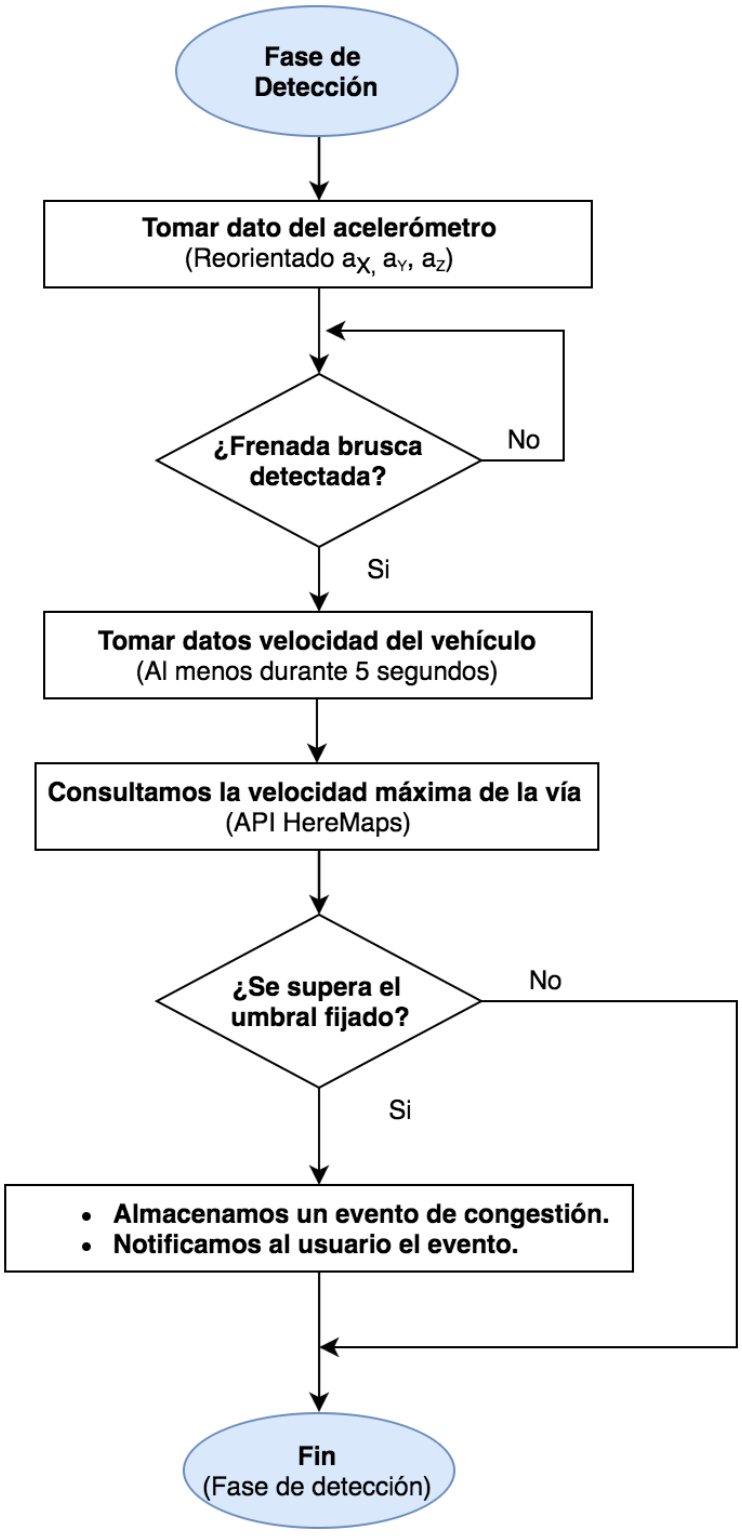

Figura 4.16: Diagrama para la fase de detección.

4.3. Aspectos generales e interfaz de usuario

Como una aplicación creada para el desarrollo y prueba, de un algoritmo alternativo de detección de la congestión vehicular, de forma general está pensada para una implementación en una entidad mayor. Su estructura provee los mecanismos necesarios para dicha integración, en un entorno de servicios de navegación.

La aplicación consta tan sólo de una actividad, que está formada por dos partes, la interfaz de usuario, y la lógica interna de operaciones. Su programación busca la mayor simplicidad, facilitando al desarrollador tener un entorno de pruebas cómodo. Esto le permite determinar la eficacia del algoritmo desarrollado.

En cuanto a su interfaz de usuario, podemos diferenciar las distintas partes comentadas en el desarrollo de la APP. A continuación, algunas imágenes que nos aclaran las partes de la propia interfaz.

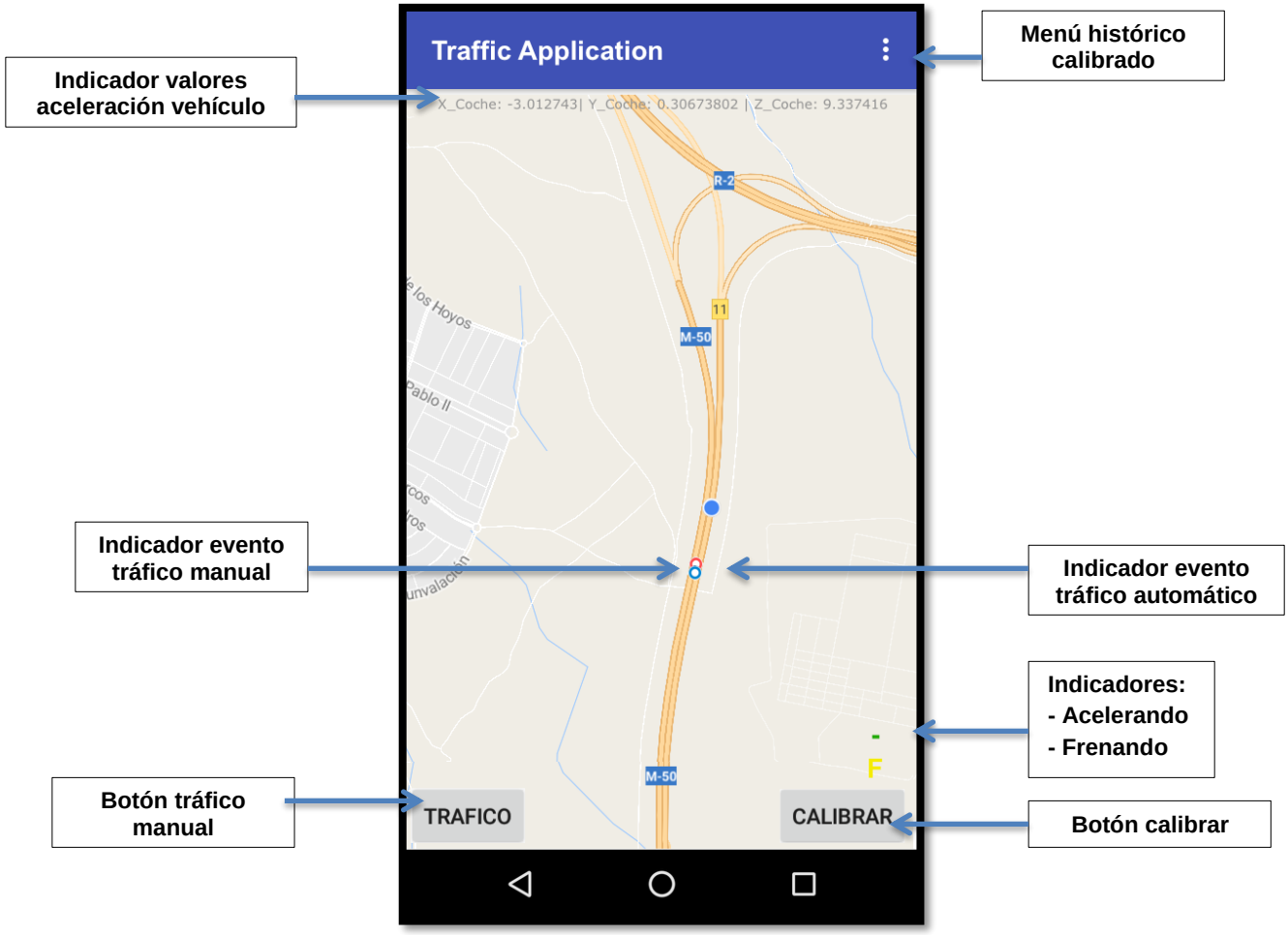

Figura 4.17: Detalle de la Interfaz de usuario.

Indicador evento tráfico automático: Indica la existencia o no de congestión vehicular, tras un evento de frenado.
Indicador evento tráfico manual: Indica la existencia o no de congestión vehicular, tras un evento manual.
Indicador valores aceleración vehículo: Indican el valor de aceleración para cada uno de los ejes del coche, a_X, a_Y, a_Z.
Botón calibrar: Inicia una nueva calibración, y permite avanzar en las diferentes fases de la misma.
Indicadores: Indica, siempre que la aplicación esté calibrada, si el vehículo se encuentra acelerando o frenando.
Menú histórico calibrado: Permite desplegar un menú como veremos en la Figura 4.18.
Botón tráfico manual: Permite al usuario indicar de forma manual la existencia de un evento de congestión vehicular.
Cargar calibrado: Permite cargar un calibrado que haya sido guardado con anterioridad.
Guardar calibrado: Permite guardar el calibrado actual de la aplicación para una futura carga.

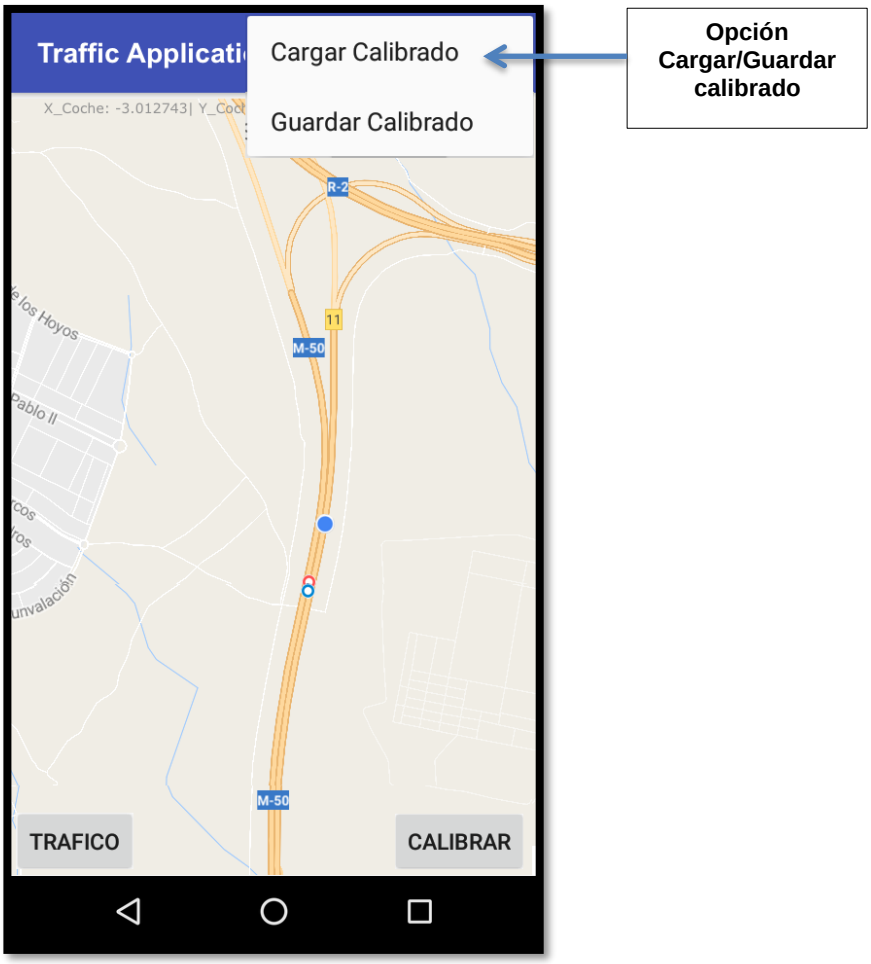

Figura 4.18: Detalle de la interfaz de usuario, cargar/guardar calibrado.

4.4. Ciclo de vida Android

En el estado del arte, se habla sobre Android y su facilidad de programación. Esta pequeña sección, ilustrará de forma rápida cómo se han aprovechado las funciones del ciclo de vida, pre-programadas en Android, para la correcta implementación del sistema.

Método	Uso
`onCreate()`	- Inicializa los aspectos de la interfaz de usuario. - Inicializa las variables globales. - Inicia los escuchadores de eventos (botones).
`onResume()`	- Muestra un cuadro para la solicitud de permiso de uso de sensores. - Apertura de ficheros para el almacenamiento de datos.
`onStop()`	- Cierre de los ficheros para el almacenamiento de datos.
`onPause()`	- Desconexión de los sensores.

Tabla 4-1: Implementación del ciclo de vida.

4.5. Actividades y clases implementadas

La APP desarrollada cuenta con una actividad principal. Para el manejo de todos los métodos, se han implementado tres clases diferentes y una actividad.

4.5.1. Actividad `main`

Es la actividad principal, donde se inicia la aplicación. Contiene todas las inicializaciones de variables, de distintos objetos, y de aquellos sensores que se deseen utilizar. Además, creará la interfaz gráfica e inicializará sus partes, así como sus escuchadores. A parte de los métodos indicados en la tabla 4-1, se implementan los siguientes métodos.

Método	Uso
`onCreateOptionsMenu()`	Crea un menú de opciones desplegable.
`onOptionsItemSelected()`	Rellena las distintas opciones del menú desplegable, además establece el vínculo con la función a desarrollar por cada opción.
`onSensorChanged()`	Establece las acciones a realizar cuando se detecte un cambio en el sensor del acelerómetro.

Tabla 4-2: Métodos de la clase `main`.

4.5.2. Clase `HereMaps`

En esta clase se crearán todos los métodos necesarios para el manejo de la API de Here WeGo. Esta clase extiende a la clase `AsyncTask`, puesto que para realizar una petición HTTP, necesitamos de una tarea asíncrona.

Método	Uso
`doInBackground()`	Permite realizar una acción en segundo plano, mientras la ejecución del hilo principal sigue su curso.
`makeCall()`	Realiza la petición HTTP. Además, una vez recibida la respuesta, se encarga de extraer los datos que necesitamos (velocidad máxima de la vía).

Tabla 4-3: Métodos de la clase `HereMaps`.

4.5.3. Clase `MyApplication`[1]

Se trata de una clase que extiende de la clase `Application`. Esta clase permite la creación de variables globales para toda la aplicación. Es una forma de agruparlas, y poder acceder a ellas de forma segura desde cualquier punto de la ejecución.

Todos los métodos implementados en esta clase, son los get y set de las variables definidas. Estos métodos permitirán la lectura y escritura de dichas variables globales.

4.5.4. Clase `Operaciones`[2]

En esta clase se agrupa el núcleo del algoritmo de detección. Podemos encontrar métodos para el calibrado del acelerómetro, para el cálculo de matrices, operaciones con vectores, etc.

[1] La clase descrita en esta sección, forma parte de las realizadas de forma común (Ver Nota al lector) [43], con el fin de facilitar el desarrollo de una base común: La reorientación del acelerómetro.
[2] La clase descrita en esta sección, forma parte de las realizadas de forma común (Ver Nota al lector) [43], con el fin de facilitar el desarrollo de una base común: La reorientación del acelerómetro.

Método	Uso
`calibradoReposo()`	Es el método que calibra el sensor acelerómetro en reposo. Calculará los ángulos Φ_{pre} y θ_{inc}, para su posterior orientación virtual.
`calibradoMovimiento()`	Es el método que calibra el sensor acelerómetro en movimiento. Calculará el ángulo ψ_{post}, para su posterior orientación virtual.
`calcularMatrizRotacion()`	Este método calcula la matriz de rotación, a partir de los valores de ψ_{post}, Φ_{pre} y θ_{inc}.
`calcularVectorMediana()`	Este método calcula el vector mediana para una lista de valores dada.
`calcularMediana()`	Este método calcula la mediana de un array de valores.
`ordenarArray()`	Este método ordena de mayor a menor los valores contenidos por un array.
`productoMatrizVectorColumna()`	Este método realiza el cálculo de la multiplicación de una matriz por un vector columna, para dar como resultado otro vector.
`productoVectorFilaMatriz()`	Este método realiza el cálculo de la multiplicación de un vector fila por una matriz, para dar como resultado otro vector.
`producto()`	Este método realiza la multiplicación entre dos matrices dadas.
`normalizarVector()`	Este método realiza la normalización de un vector dado.
`calcularMedia()`	Este método realiza la media de una lista de valores dada.
`escribirFichero()`	Este método permite la escritura en un fichero.
`cargarCalibrado()`	Este método es el llamado al pulsar la opción cargar calibrado.
`guardarCalibrado()`	Este método es el llamado al pulsar la opción guardar calibrado.

Tabla 4-4: Métodos de la clase `Operaciones`.

5. Evaluación y resultados

En este capítulo se pondrá a prueba la aplicación desarrollada, comprobando el correcto funcionamiento de cada una de sus partes. Se analizarán los resultados obtenidos, y se concluirá si es correcta o no.

5.1. Entorno de pruebas

Para la realización de pruebas en la aplicación, tanto en su desarrollo como en su finalización se han utilizado dos tipos de entornos distintos. Por un lado un entorno virtual y por otro lado un entorno físico.

- **Entorno Virtual:** Android Studio provee una herramienta llamada Android Virtual Device (AVD). Durante las pruebas se ha utilizado este entorno con una simulación del dispositivo Nexus 5 en la versión de API número 23.
- **Entorno Físico:** Los dispositivos físicos utilizados para la realización de pruebas de la aplicación han sido:
  - BQ Aquaris 4.5.
  - Samsung A5.

5.2. Pruebas

Para la descripción de las diferentes pruebas a realizar, se ha construido una tabla con la siguiente información:

Identificador:	PS-XX
Nombre:	
Descripción:	
Pasos:	
Resultado:	

Tabla 5-1: Ejemplo tabla pruebas del sistema (PS).

- **Identificador**: Para la identificación de cada prueba del sistema se le asignará un número de identificador único. Este número empezará por PS (prueba del sistema) seguido de un número.
- **Nombre:** Titular que resume la prueba.
- **Descripción:** Breve descripción sobre el objetivo de la prueba.
- **Pasos:** Pasos a realizar para verificar y reproducir la prueba.
- **Resultado:** Resultado de la prueba, puede ser Correcta o Incorrecta.

Se ha decidido realizar una clasificación en distintos tipos de prueba para facilitar su identificación y lectura.

5.2.1. Pruebas de interfaz

En esta sección se incluirán todas las pruebas sobre la interfaz. Estas pruebas indicarán la aparición o no de los elementos a probar.

Identificador:	PS-01
Nombre:	Mensaje de permisos (GPS y almacenamiento).
Descripción:	Aparición del mensaje de permisos al entrar por primera vez en la aplicación, o si los permisos de esta han sido retirados.
Pasos:	1. Desactivar los permisos para el sensor GPS y el almacenamiento. 2. Abrir la aplicación. 3. Comprobar que el mensaje aparece.
Resultado:	✓ Correcto.

Tabla 5-2: PS-01: Mensaje de permisos (GPS y almacenamiento).

Identificador:	PS-02
Nombre:	Mensaje de tráfico.
Descripción:	El mensaje de tráfico se muestra.
Pasos:	1. Abrir la aplicación. 2. Calibrar de forma correcta. 3. Detectar un evento de frenado. 4. Se mostrará si el evento es de congestión o no.
Resultado:	✓ Correcto.

Tabla 5-3: PS-02: Mensaje de tráfico.

Identificador:	PS-03
Nombre:	Mensaje con los valores de aceleración a_X, a_Y, a_Z.
Descripción:	Se mostrarán los valores de a_X, a_Y, a_Z una vez el sensor esté reorientado.
Pasos:	1. Abrir la aplicación. 2. Calibrar de forma correcta. 3. Verificar que aparecen los valores.
Resultado:	✓ Correcto.

Tabla 5-4: PS-03: Mensaje con los valores de aceleración a_X, a_Y, a_Z.

Identificador:	PS-04
Nombre:	Mensaje acelerando/frenando
Descripción:	Se mostrará el mensaje acelerando/frenando
Pasos:	1. Abrir la aplicación. 2. Calibrar de forma correcta. 3. Detectar evento aceleración/frenado.
Resultado:	✓ Correcto.

Tabla 5-5: PS-04: Mensaje acelerando/frenando.

Identificador:	PS-05
Nombre:	Botón de tráfico.
Descripción:	Aparecerá un botón con el texto 'TRAFICO' y podrá ser pulsado.
Pasos:	1. Abrir la aplicación. 2. Comprobar que aparece el botón.
Resultado:	✓ Correcto.

Tabla 5-6: PS-05: Botón de tráfico.

Identificador:	PS-06
Nombre:	Botón de calibrar.
Descripción:	Aparecerá un botón con el texto 'CALIBRAR' y podrá ser pulsado.
Pasos:	1. Abrir la aplicación. 2. Comprobar que aparece el botón.
Resultado:	✓ Correcto.

Tabla 5-7: PS-06: Botón de calibrar.

Identificador:	PS-07
Nombre:	Botón de cargar.
Descripción:	Aparecerá un botón en el menú desplegable con el texto 'Cargar Calibrado' y podrá ser pulsado.
Pasos:	1. Abrir la aplicación. 2. Desplegar el menú. 3. Comprobar que aparece el botón.
Resultado:	✓ Correcto.

Tabla 5-8: PS-07: Botón de cargar.

Identificador:	PS-08
Nombre:	Botón de guardar.
Descripción:	Aparecerá un botón en el menú desplegable con el texto 'Guardar Calibrado' y podrá ser pulsado.
Pasos:	1. Abrir la aplicación. 2. Desplegar el menú. 3. Comprobar que aparece el botón.
Resultado:	✓ Correcto.

Tabla 5-9: PS-08: Botón guardar.

5.2.2. Pruebas de funcionamiento

En esta sección se incluirán todas las pruebas sobre el correcto funcionamiento de todos aquellos elementos que aparecen en la interfaz.

Identificador:	PS-09
Nombre:	Funcionamiento mensaje de permisos (GPS y almacenamiento).
Descripción:	Al pulsar aceptar, se accederá a la aplicación. Si por el contrario se pulsa denegar, será imposible acceder a ella.
Pasos:	1. Desactivar los permisos para el sensor GPS y el almacenamiento. 2. Abrir la aplicación. 3. Aceptar/Denegar.
Resultado:	✓ Correcto.

Tabla 5-10: PS-09: Funcionamiento mensaje de permisos (GPS y almacenamiento).

Identificador:	PS-10
Nombre:	Funcionamiento mensaje de tráfico.
Descripción:	El mensaje de tráfico se muestra de forma correcta. Es decir, que si se detecta trafico se mostrará el mensaje TRAFFIC, seguido de la velocidad del vehículo y la máxima de la vía.
Pasos:	1. Abrir la aplicación. 2. Calibrar de forma correcta. 3. Detectar un evento de frenado. 4. Se mostrará si el evento es de congestión o no. 5. Comprobar que ese mensaje es correcto.
Resultado:	✓ Correcto.

Tabla 5-11: PS-10: Funcionamiento mensaje de tráfico.

Identificador:	PS-11
Nombre:	Funcionamiento mensaje con los valores de aceleración a_X, a_Y, a_Z.
Descripción:	Los valores a_X, a_Y, a_Z se corresponderán con los de un sensor correctamente orientado.
Pasos:	1. Abrir la aplicación. 2. Calibrar de forma correcta. 3. Verificar que aparecen los valores son correctos.
Resultado:	✓ Correcto.

Tabla 5-12: PS-11: Funcionamiento mensaje con los valores de aceleración a_X, a_Y, a_Z.

Identificador:	PS-12
Nombre:	Funcionamiento mensaje acelerando/frenando.
Descripción:	Se mostrará el mensaje acelerando/frenando, cuando se produzca una aceleración o frenado de suficiente intensidad.
Pasos:	1. Abrir la aplicación. 2. Calibrar de forma correcta. 3. Detectar evento aceleración/frenado. 4. Comprobar que es suficientemente intenso.
Resultado:	✓ Correcto.

Tabla 5-13: PS-12: Funcionamiento mensaje acelerando/frenando.

Identificador:	PS-13
Nombre:	Funcionamiento botón de tráfico.
Descripción:	Al pulsar el botón de tráfico se creará un evento manual de cogestión vehicular.
Pasos:	1. Abrir la aplicación. 2. Calibrar de forma correcta. 3. Pulsar el botón detección manual de tráfico. 4. Comprobar el registro de eventos.
Resultado:	✓ Correcto.

Tabla 5-14: PS-13: Funcionamiento botón de tráfico.

Identificador:	PS-14
Nombre:	Funcionamiento botón de calibrar.
Descripción:	Al pulsar el botón calibrar, se producirá el calibrado en las dos fases definidas, en reposo y en movimiento.
Pasos:	1. Abrir la aplicación. 2. Pulsar el botón de calibrar.
Resultado:	✓ Correcto.

Tabla 5-15: PS-14: Funcionamiento botón de calibrar.

Identificador:	PS-15
Nombre:	Funcionamiento botón de cargar.
Descripción:	Al pulsar el botón cargar, se cargará una configuración de calibrado previamente guardada.
Pasos:	1. Abrir la aplicación. 2. Desplegar el menú 3. Pulsar el botón cargar 4. Se cargará una configuración guardada.
Resultado:	✓ Correcto.

Tabla 5-16: PS-15: Funcionamiento del botón de cargar.

Identificador:	PS-16
Nombre:	Funcionamiento botón de guardar.
Descripción:	Al pulsar el botón guardar, se guardará la configuración actual de calibrado.
Pasos:	1. Abrir la aplicación. 2. Calibrar aplicación. 3. Desplegar el menú. 4. Pulsar el botón guardar. 5. Se guardará la configuración.
Resultado:	✓ Correcto.

Tabla 5-17: PS-16: Funcionamiento botón guardar.

5.2.3. Pruebas de fiabilidad

En esta sección se definirán las pruebas creadas para comprobar la fiabilidad de la aplicación en su objetivo principal: detectar eventos de congestión vehicular en las vías.

Identificador:	PS-17
Nombre:	Detección de tráfico manual.
Descripción:	En esta prueba se comprobará si los eventos de congestión en el tráfico manuales, son almacenados de forma correcta.
Pasos:	1. Abrir la aplicación. 2. Calibrar aplicación. 3. Desplegar el menú. 4. Pulsar el botón tráfico. 5. Comprobar que se almacena el evento, y se muestra al usuario.
Resultado:	✓ Correcto.

Tabla 5-18: PS-17: Fiabilidad detección de tráfico manual.

Identificador:	PS-18
Nombre:	Correcto calibrado.
Descripción:	En esta prueba se comprobará si el calibrado se ha realizado de forma correcta.
Pasos:	1. Colocar el dispositivo móvil en una posición fija. 2. Abrir la aplicación. 3. Calibrar aplicación. 4. Comprobar si los ejes del calibrado se corresponden con los ejes del coche. Para ello usamos los valores mostrados.
Resultado:	✓ Correcto.

Tabla 5-19: PS-18: Fiabilidad del calibrado.

Identificador:	PS-19
Nombre:	Formato de almacenamiento de eventos
Descripción:	En esta prueba se comprobará el correcto formato en el almacenamiento de los eventos producidos.
Pasos:	1. Abrir la aplicación. 2. Calibrar aplicación. 3. Desplegar el menú. 4. Pulsar el botón tráfico o esperar evento automático 5. Cerrar la aplicación. 6. Comprobar log de eventos y su formato (compatible con Matlab)
Resultado:	✓ Correcto.

Tabla 5-20: PS-19: Fiabilidad detección de tráfico manual.

A continuación, adjuntamos dos imágenes relacionadas con PS-19, y su comprobación:

```
1
2  traficoAuto = [9.168232,1468214748402,40.33179666666667-3.77272;
3  9.013816,1468214781410,40.33293666666667-3.7713066666666664;
4  1.2862909,1468214819209,40.33387333333334-3.769333333333334;
5  6.406233,1468214861794,40.335921666666664-3.769403333333333;
6  12.818128,1468214892436,40.33749666666667-3.7699033333333336;
7  4.3869777,1468215006407,40.34607166666667-3.764226666666666;
8  13.813086,1468216066593,40.429343333333335-3.5004616666666664;
9  17.856743,1468216156036,40.442748333333334-3.5133333333333336;
10 18.346245,1468216201016,40.447630000000004-3.5190733333333335;
11 18.366318,1468216241803,40.44526833333334-3.5258100000000003;
12 0.0,1468216278401,40.44515-3.5283599999999997;
13 0.0,1468216358196,40.445135-3.5291433333333337;
14 1468216359941, 0, 0];
```

Figura 5.1: Ejemplo del formato log de eventos para el tráfico automático.

```
1
2  trafico = [17.75174,1468216051944,40.427440000000004-3.4995016666666667;
3  0.0,1468216274764,40.44515-3.5283599999999997;
4  1468216359941, 0, 0];
```

Figura 5.2: Ejemplo del formato log de eventos para el tráfico manual.

Identificador:	PS-20
Nombre:	Correcta detección tráfico automático
Descripción:	En esta prueba se comprobará el correcto funcionamiento del algoritmo automático.
Pasos:	1. Abrir la aplicación. 2. Calibrar aplicación. 3. Esperar eventos automáticos. 4. Registrar eventos manuales. 5. Comprobar que se almacena el evento, y se muestra al usuario. 6. Comprobar los datos almacenados se corresponden con otras fuentes.
Resultado:	✓ Correcto.

Tabla 5-21: PS-20: Fiabilidad detección de tráfico manual.

5.2.4. Comprobando el funcionamiento del algoritmo

Basándonos en la PS-20, hemos probado la fiabilidad del algoritmo. Puesto que este algoritmo es el corazón del proyecto, se dedicará esta sección a explicar más a fondo los métodos de prueba utilizados, demostrando su correcto desempeño.

Para comprobar el correcto funcionamiento se han registrado datos durante varios días alternos. A continuación, explicaremos la ruta seguida uno de esos días y expondremos los datos.

Las pruebas finales de la aplicación se han realizado en un entorno real. La siguiente figura corresponde a la ruta desde el domicilio del desarrollador situado en Leganés, hasta su centro de trabajo situado en las inmediaciones del aeropuerto de Barajas.

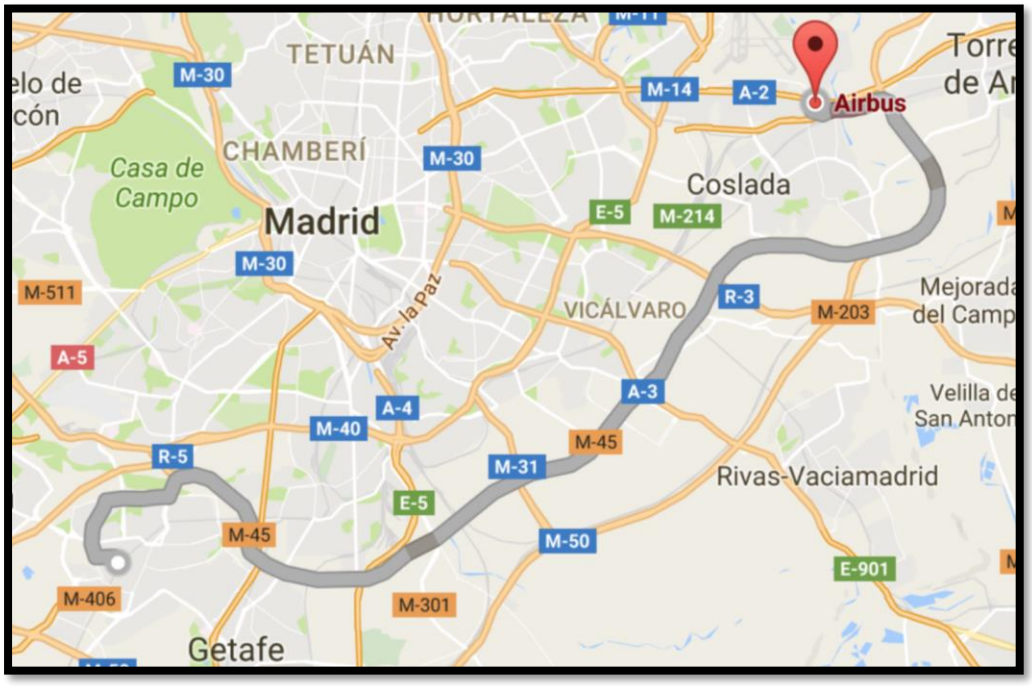

Figura 5.3: Detalle de la ruta Leganés - Barajas. (Fuente Google Maps)

Los detalles de la ruta son:

- **Hora aproximada de salida:** 7:25.
- **Hora aproximada de llegada:** 8:00.
- **Origen:** Leganés (Madrid).
- **Destino:** Avenida de Aragón, 404, Madrid.
- **Vías principales de la ruta:** M-45, M-50, M-21.

Durante el desarrollo de la ruta se tomaron datos a través de los diferentes métodos desarrollados. Por un lado, el copiloto realizaba registros manuales al detectar congestión vehicular de forma visual. Por otro lado, el algoritmo implementado, hacía lo mismo de forma automática. Por último, para tener un registro del funcionamiento, se registraba la velocidad instantánea del vehículo, y la máxima de la vía. A continuación, los datos y el resultado:

Velocidad [m/s]	Tiempo absoluto [ms]	Latitud [º]	Longitud [º]
17.75174	1468216051944 11/07/2016 –7:47	40.427440000000004	-3.4995016666666667
0.0	1468216274764 11/07/2016 –7:51	40.44515	-3.5283599999999997

Tabla 5-22: Datos obtenidos manualmente.

Velocidad [m/s]	Tiempo absoluto [ms]	Latitud [º]	Longitud [º]
13.813086	1468216066593 11/07/2016 –7:47	40.429343333333335	-3.5004616666666664
17.856743	1468216156036 11/07/2016 –7:49	40.442748333333334	-3.5133333333333336
18.346245	1468216201016 11/07/2016 –7:50	40.447630000000004	-3.5190733333333335
18.366318	1468216241803 11/07/2016 –7:50	40.44526833333334	-3.5258100000000003
0.0	1468216278401 11/07/2016 –7:51	40.44515	-3.5283599999999997
0.0	1468216358196 11/07/2016 –7:52	40.445135	-3.5291433333333337

Tabla 5-23: Datos obtenidos automáticamente.

Si representamos en un mapa todos los puntos, comprobamos que de forma general coinciden en dos diferentes áreas. Un área de la M-50 y un área en la salida de la M-21. Se observa también la repetición de varios puntos en el algoritmo automático. Esto puede ser debido al patrón que se repite cuando la congestión es tan fuerte, que el tráfico permanece inmóvil. La repetición podría deberse al proceso de arrancar/frenar que se produce en lo que comúnmente se denomina ‘atasco’.

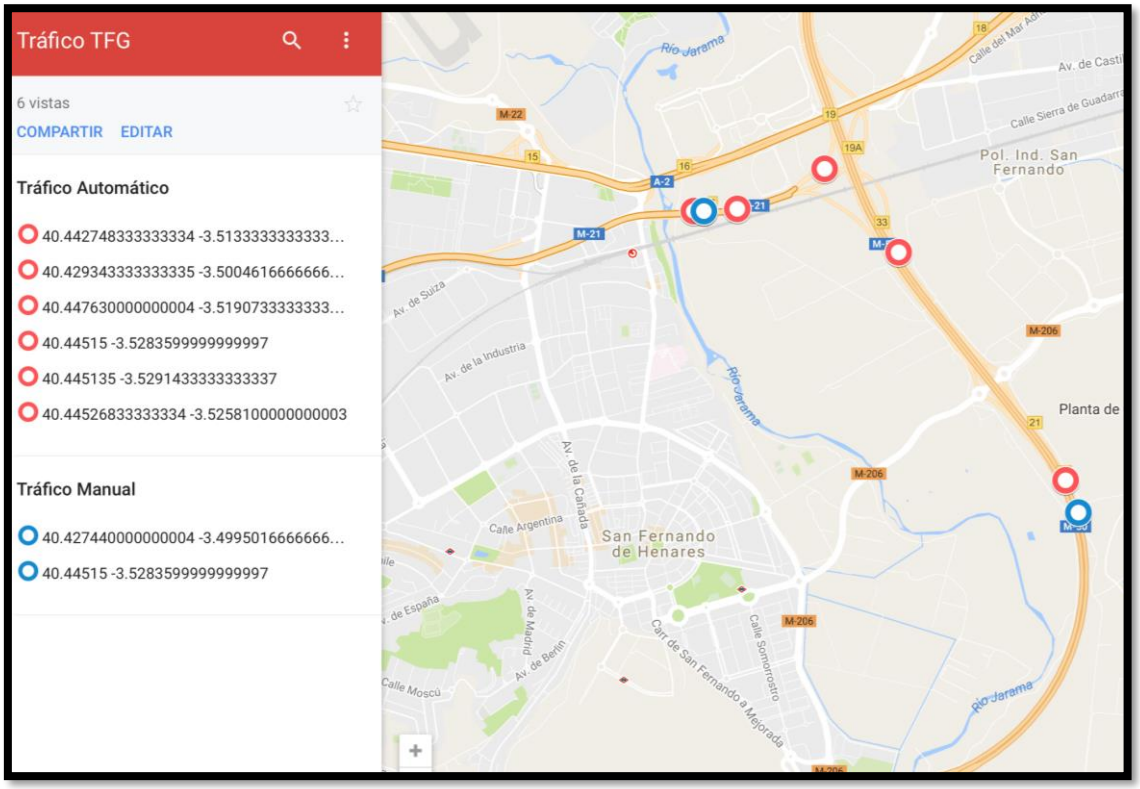

Figura 5.4: Representación de los datos automáticos y manuales recogidos. (Fuente Google Maps) [28]

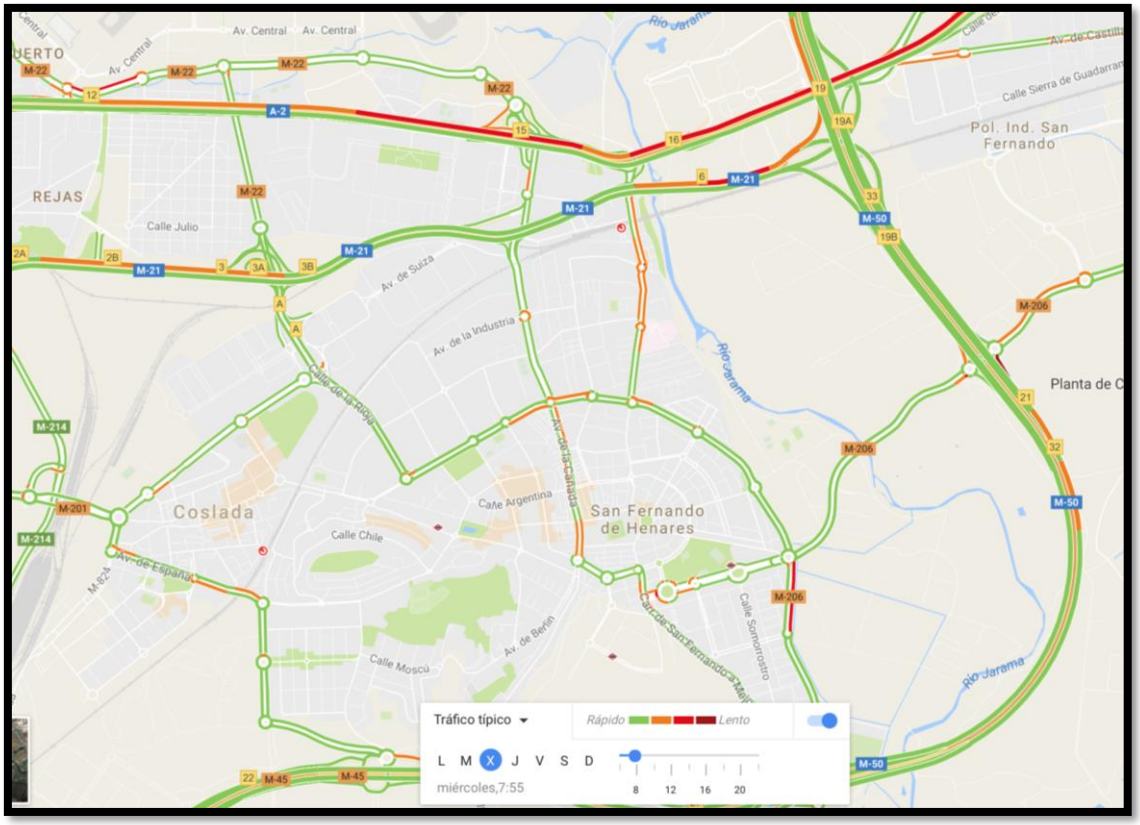

Figura 5.5: Histórico de tráfico en la web de Google Maps. (Fuente Google Maps)

Con la ayuda de Google Maps, y su histórico de tráfico típico, se ha podido obtener un gráfico de la zona, con el tráfico típico. Para ello se ha seleccionado un día laboral, con una hora similar a la de nuestros datos (7:55). El resultado en la figura 5.5.

Si superponemos ambas figuras, y prestamos atención a las dos zonas marcadas por nuestra aplicación, descubrimos lo siguiente. (Se debe tener en cuenta que el sentido de la ruta es ascendente como ya vimos en la figura 5.3, por lo que la marca de tráfico que nos aplica se sitúa en el carril derecho).

Zona 1 – M50: La zona 1 (figura 5.6), corresponde a la zona de intersección, donde M-45 y M-50 se unen. Como podemos observar la zona detectada corresponde con una marca naranja del tráfico histórico. Lo que junto con la marca azul del usuario indica que la marca roja de evento automático es correcta.

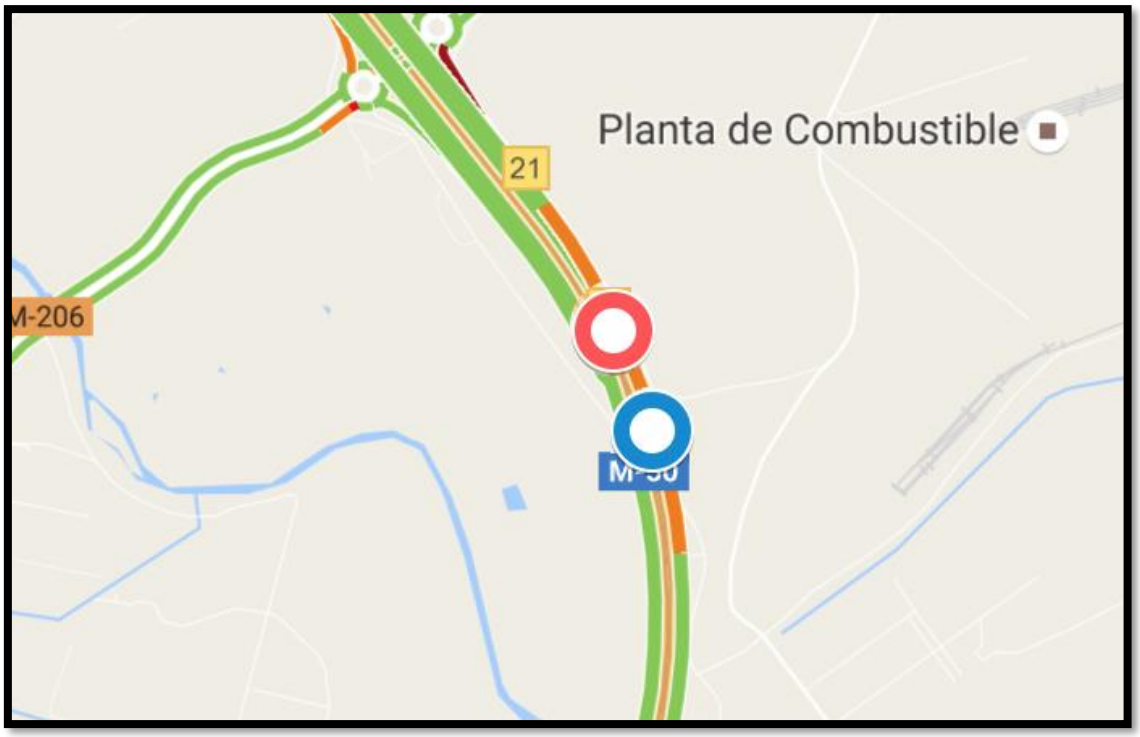

Figura 5.6: Detalle de los mapas superpuestos para la Zona 1 - M-50. (Fuente Google Maps)

Zona 2 – M21: La zona 2 (figura 5.7), corresponde a la zona de intersección, donde los vehículos procedentes de la M-50 y de la A-2 confluyen para dar inicio a la carretera M-21. Además, se da la circunstancia de encontrarse en el mismo punto la salida a una zona industrial, donde a la hora de la toma de datos hay gran número de vehículos. Como podemos observar la zona detectada corresponde con una marca naranja, e incluso en algunos tramos roja, correspondiente a

tráfico histórico. Como ocurriera en la Zona 1, confirma los eventos detectados (rojo), junto con la información manual (azul) . La detección vuelve a ser correcta.

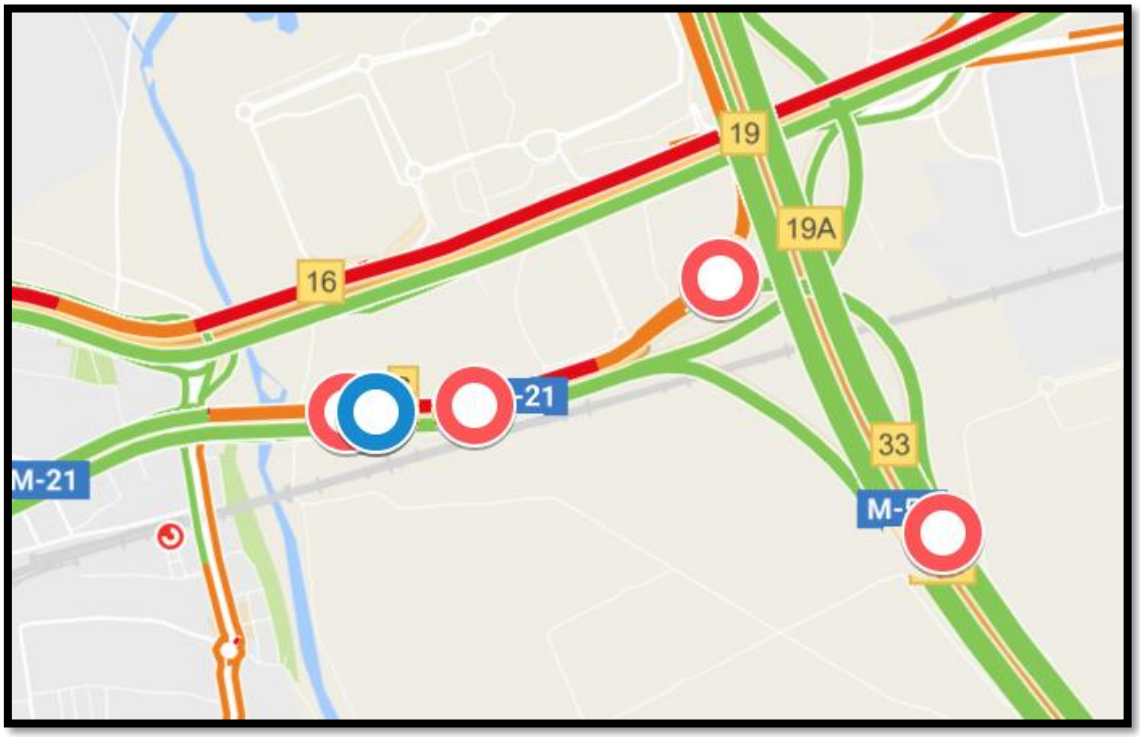

Figura 5.7: Detalle de los mapas superpuestos para la Zona 2 – M-21. (Fuente Google Maps)

6. Marco regulador

En el desarrollo de este capítulo se realizará un repaso de la legislación vigente en España y en el marco de Europa, que afecta a este TFG. Se trata de poner de manifiesto las leyes de obligado cumplimiento, en caso de querer lanzar la aplicación al público general.

Constitución Española[21]: Ratificada en referéndum el 6 de diciembre de 1978, se trata de la norma suprema del ordenamiento jurídico español. Entre los artículos que afectan al desarrollo de este TFG encontramos:

Artículo 18:

1. **Se garantiza el derecho al honor, a la intimidad personal y familiar y a la propia imagen.**
2. El domicilio es inviolable. Ninguna entrada o registro podrá hacerse en él sin consentimiento del titular o resolución judicial, salvo en caso de flagrante delito.
3. **Se garantiza el secreto de las comunicaciones, y en especial de las postales, telegráficas y telefónicas, salvo resolución judicial.**
4. **La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos.**

Se deberá tener esto en cuenta en el desarrollo de aplicaciones móviles, garantizando el secreto de las comunicaciones del usuario, así como la preservación de su honor e intimidad.

Ley de Propiedad Intelectual[22]**:** Legisla lo relativo a la propiedad intelectual, que es el conjunto de derechos que corresponden a los autores y a otros artistas, respecto de las obras y prestaciones fruto de su creación. En el desarrollo de su texto, la Ley de Propiedad Intelectual distingue tipos de derechos: derechos morales y derechos de carácter patrimonial.

Para el cumplimiento de lo dispuesto se establecen una serie de mecanismos de protección en su Libro III, Título I, ante cualquier tipo de infracción en los derechos.

Por tanto, cualquier desarrollo software deberá cumplir con lo dispuesto en esta ley, respetando la propiedad intelectual de otras obras, y haciendo respetar la suya propia.

Ley General de la Comunicación Audiovisual[23]**:** Legisla lo relativo a las comunicaciones y contenidos audiovisuales. Tiene por objeto la regulación y ordenación del sector, con la intención de proteger al ciudadano de posiciones dominantes de opinión, o de la restricción de acceso a contenidos universales de gran interés o valor.

Basada en la Directiva 2007/65/CE de la Unión Europea, busca establecer y perfeccionar normas que configuren un régimen básico común, garantizando el pluralismo y los derechos de los consumidores.

En su artículo 7, la Ley General de la Comunicación Audiovisual, trata los derechos de los menores de edad. Se provee un marco para la etiquetación, control y restricción de contenidos, que puede ser necesario aplicarse en software como videojuegos, reproductores de video, o aplicaciones de *streaming*. De esta forma, una buena práctica será etiquetar correctamente la aplicación para definir su público.

Ley Orgánica de Protección de Datos de Carácter Personal[24]**:** Tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, especialmente de su honor e intimidad personal y familiar.

Se aplicará a todos los datos de carácter personal que se encuentren registrados en soporte físico, y que sean susceptibles de tratamiento y uso posterior por los sectores público y privado. La definición de datos de carácter personal es: cualquier información concerniente a personas físicas identificadas o identificables.

Para el desarrollo de una aplicación móvil en cumplimiento con esta ley se deberá:

- Informar a los usuarios sobre sus derechos a ejercer para la protección de sus datos.
- Acotar el propósito por el cual se recoge información.
- Contar con el consentimiento previo del usuario.

Principio libre circulación: Definido en el Artículo 13.1 de la Declaración Universal de los Derechos Humanos de Naciones Unidas, y una de las bases de la Unión Europea. En el caso europeo, se regula el derecho de las personas a circular y residir libremente dentro de la Unión Europea.

La supresión de las fronteras entre los países firmantes del tratado Schengen, posibilita la circulación de vehículos.

Reglamento General de Circulación[25]**:** Supone la base para el desarrollo de la Ley sobre tráfico, circulación de vehículos a motor y seguridad vial.

En su Artículo 18 sobre "Otras obligaciones del conductor" se ordena:

1. **El conductor de un vehículo está obligado a mantener** su propia libertad de movimientos, **el campo necesario de visión y la atención permanente a la conducción**, que garanticen su propia seguridad, la del resto de los ocupantes del vehículo y la de los demás usuarios de la vía. A estos efectos deberá cuidar especialmente de mantener la posición adecuada y que la mantengan el resto de los pasajeros, y la adecuada colocación de los objetos o animales transportados para que no haya interferencia entre el conductor y cualquiera de ellos.

   Se considera incompatible con la obligatoria atención permanente a la conducción el uso por el conductor con el vehículo en movimiento de dispositivos tales como pantallas con acceso a internet, monitores de televisión y reproductores de vídeo o DVD. Se exceptúan, a estos efectos, el uso de monitores que estén a la vista del conductor y cuya utilización sea necesaria para la visión de acceso o bajada de peatones o para la visión en vehículos con cámara de maniobras traseras, así como el dispositivo GPS.

2. **Queda prohibido conducir y utilizar cascos o auriculares conectados a aparatos receptores o reproductores de sonido**, excepto durante la correspondiente enseñanza y la realización de las pruebas de aptitud en circuito abierto para la obtención del permiso de conducción de motocicletas de dos ruedas cuando así lo exija el Reglamento General de Conductores.

   Se prohíbe la utilización durante la conducción de dispositivos de telefonía móvil y cualquier otro medio o sistema de comunicación, excepto cuando el desarrollo de la comunicación tenga lugar sin emplear las manos ni usar cascos, auriculares o instrumentos similares.

English version available for this chapter.

7. Conclusión

En este capítulo se presentarán las conclusiones extraídas de la realización y desarrollo de este TFG. Para ello se revisarán los objetivos alcanzados, y se establecerán una serie de líneas de trabajo futuras.

La oportunidad de realizar este proyecto, ha supuesto en reto tanto personal como académico/profesional. El empeño y dedicación durante su desarrollo, ha supuesto la adquisición de una nueva perspectiva para la resolución de problemas, que será de gran utilidad en proyectos futuros.

Cabe destacar que la solución propuesta a este proyecto, no es única, ni tampoco es la más completa, si bien es aquella que mejor encajaba con los recursos y conocimientos disponibles. Como hemos observado durante el desarrollo de esta memoria, se han barajado diferentes alternativas en los diferentes ámbitos del problema:

- Diferentes sistemas operativos, eligiendo finalmente Android.
- Diferentes sensores, eligiendo finalmente el acelerómetro.
- Diferentes artículos, que daban solución al calibrado del acelerómetro.
- Diferentes API's para la obtención de información de tráfico, eligiendo finalmente Here WeGo.

Todas estas decisiones han sido debidamente fundamentadas en el trascurso de este documento.

7.1. Objetivos alcanzados

Durante el inicio del proyecto se establecieron una serie de objetivos. Su consecución o no, resultará clave para medir el éxito del mismo.

7.1.1. Objetivos del proyecto

Objetivo	Consecución
Principal: Desarrollo de una aplicación Android basada en un algoritmo, que permita la detección del tráfico.	✓ Se ha desarrollado una aplicación Android que detecta con éxito las situaciones de congestión vehicular.
Secundarios: Respetar al máximo la privacidad del usuario, informando de la ubicación de los eventos única y exclusivamente.	✓ Durante el desarrollo, se ha tenido en cuenta la privacidad del usuario, con la realización de un estudio del marco regulador.

La detección de los eventos se realizará de forma local, es decir, en el propio dispositivo.	✓ La detección se realiza de forma local, mediante el acelerómetro. Sólo se envía información de los eventos de tráfico detectados. **NUNCA** de la ruta completa del usuario
La gestión de recursos deberá ser eficaz.	✓ El control y selección de los datos a enviar, permiten tanto un ahorro de batería, como un ahorro en los datos.
Importancia del algoritmo por encima de la APP	✓ La aplicación desarrollada, tiene como objetivo la prueba y verificación de dicho algoritmo.

Tabla 7-1: Objetivos del proyecto alcanzados.

7.1.2. Objetivos personales

Objetivo	Consecución
Aprender a desarrollar en aplicaciones móviles	✓ Se ha desarrollado una aplicación Android con éxito.
Estudiar y utilizar metodologías ágiles.	✓ Se ha desarrollado un proyecto mediante una metodología ágil con éxito.

Tabla 7-2: Objetivos personales alcanzados.

7.2. Líneas futuras de trabajo

Las posibilidades de trabajo y expansión para este desarrollo son altas a medio y corto plazo.

- Gracias a la comparación entre los datos manuales del usuario, y los datos automáticos, se podrá establecer en un futuro diferentes niveles de congestión en la vía (similar a los colores mostrados en Google).
- Su inclusión en un navegador, donde además de esta información se pueda utilizar información de terceros (ejemplo: DGT), es una de las mayores posibilidades para este desarrollo. Se podría minimizar el número de datos necesarios del usuario para ofrecer un servicio de tráfico fiable.
- Se podrá optimizar al máximo la calibración del dispositivo móvil y su acelerómetro. Con las herramientas disponibles, se puede desarrollar un automatismo para que las tareas de calibrado sean realizadas de forma autónoma, sin la necesidad de atención por parte del usuario.
- Con el registro de los eventos de tráfico en un servidor, se podrá proveer de esta información en tiempo real a otros usuarios. Incluso dentro de una aplicación de navegación, se podrán ofrecer rutas alternativas más rápidas.

Anexo A: Gestión del proyecto y presupuesto

Este anexo detallará la gestión del proyecto y la estimación de presupuesto. Se trata de un punto importante en el desarrollo del proyecto, ya que mostrará la gestión realizada, tanto de tiempo como de recursos. Para ello se detallarán las fases, se realizará un diagrama de Gantt, y por último se detallarán los costes de herramientas y personal.

Gestión del proyecto

La realización de este proyecto comenzó en el mes de Noviembre de 2015, y su conclusión tiene lugar en Septiembre de 2016, por lo que aproximadamente su desarrollo ha sido de 10 meses. Se debe tener en cuenta a la hora de valorar los tiempos de cada fase, que se ha compatibilizado la tarea de desarrollo junto con:

- Trabajo a media jornada (6h diarias, de lunes a viernes)
- Realización de tres asignaturas correspondientes a 18 créditos ECTS en total (unas 450h de trabajo aproximadamente).

Por lo tanto, deberá tenerse en cuenta, a la hora de visualizar el diagrama de Gantt, que la duración de las tareas en el calendario, no concuerda con el número de horas totales para cada tarea (puesto que ha habido épocas del año en que se ha trabajado más a fondo en este proyecto, que en otras). Por ello se elaborará una tabla de tareas/horas dedicadas.

Estimación inicial

Antes del inicio del proyecto, se estimaron una serie de tareas, y se dividió el número de horas utilizadas para cada una de ellas. Las tareas son aquellas definidas en el punto 1.3. Fases de desarrollo.

Tarea	Tiempo estimado [h]
1. Análisis del problema.	10
2. Formación.	20
3. Desarrollo.	200
4. Aplicación final.	50
5. Documentación.	120
TOTAL	400

Tabla A.1: Duración estimada de cada etapa.

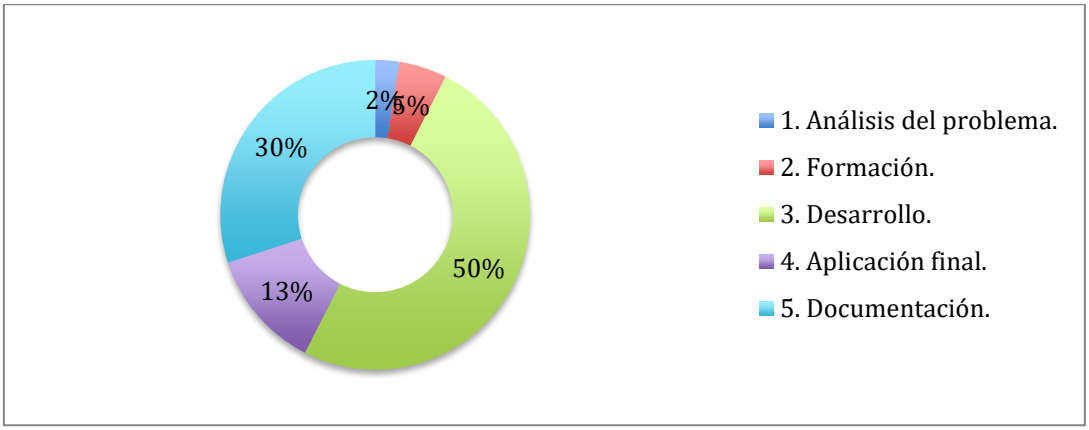

Figura A.1: Porcentaje nº de horas estimadas para las distintas tareas.

Planificación final

Con la finalización del proyecto, se ha realizado un análisis de las tareas realizadas finalmente, y se ha contabilizado el tiempo real de trabajo. A continuación, la tabla de tareas desglosadas y el recuento de horas:

Tarea	Tiempo estimado [h]
1. Análisis del problema:	(Total 10 h)
a. Investigación*:	4
b. Lectura del artículo*:	2
c. Planteamiento inicial*:	4
2. Formación:	(Total 25 h)
a. Formación teórica Android*:	10
b. Formación práctica Android*:	15
3. Desarrollo:	(Total 200 h)
a. Iteración 1: APP GPS*:	20
b. Iteración 2: APP acelerómetro*:	15
c. Iteración 3: Calibrado*:	60
d. Iteración 4: Toma de datos:	50
e. Iteración 5: Algoritmo Here WeGo:	55
4. Aplicación final:	(Total 80 h)
a. Integración diferentes partes:	50
b. Pruebas de uso real:	30
5. Análisis de datos con Matlab:	(Total 15 h)
6. Documentación:	(Total 150 h)
TOTAL	480

Tabla A.2: Desglose de tareas y su duración en nº de horas.

* Las tareas marcadas, son aquellas realizadas de forma común (Ver Nota al lector) [43], con el fin de facilitar el desarrollo de una base común, y adquirir conocimiento en la programación Android.

Además, se debe tener en cuenta que durante el desarrollo del proyecto, se han realizado varias reuniones. Entre ellas la de inicio de proyecto, varias de seguimiento y una de finalización. En estas reuniones han tomado partido, tanto el desarrollador como su tutor. A continuación, una tabla resumen:

Fecha	Reunión	Tipo
17 Septiembre 2015.	Información ofertas TFG.	Informativa.
25 Octubre 2015.	Aceptación e inicio del TFG.	Inicio del proyecto.
30 Marzo 2016.	Muestra iteraciones 1 y 2.	Seguimiento.
4 Mayo 2016.	Muestra iteraciones 3 y 4.	Seguimiento.
13 Mayo 2016.	Resolución dudas iteración 5.	Técnica.
2 Junio 2016.	Muestra APP final y análisis datos.	Seguimiento.
15 Junio 2016.	Inicio de la documentación.	Seguimiento.
8 Septiembre 2016.	Revisión documentación y cierre del proyecto.	Finalización

Tabla A-3: Resumen de las reuniones mantenidas.

Por último, como parte de la gestión, se adjunta un diagrama de Gantt para todas las tareas definidas en la tabla A-2.

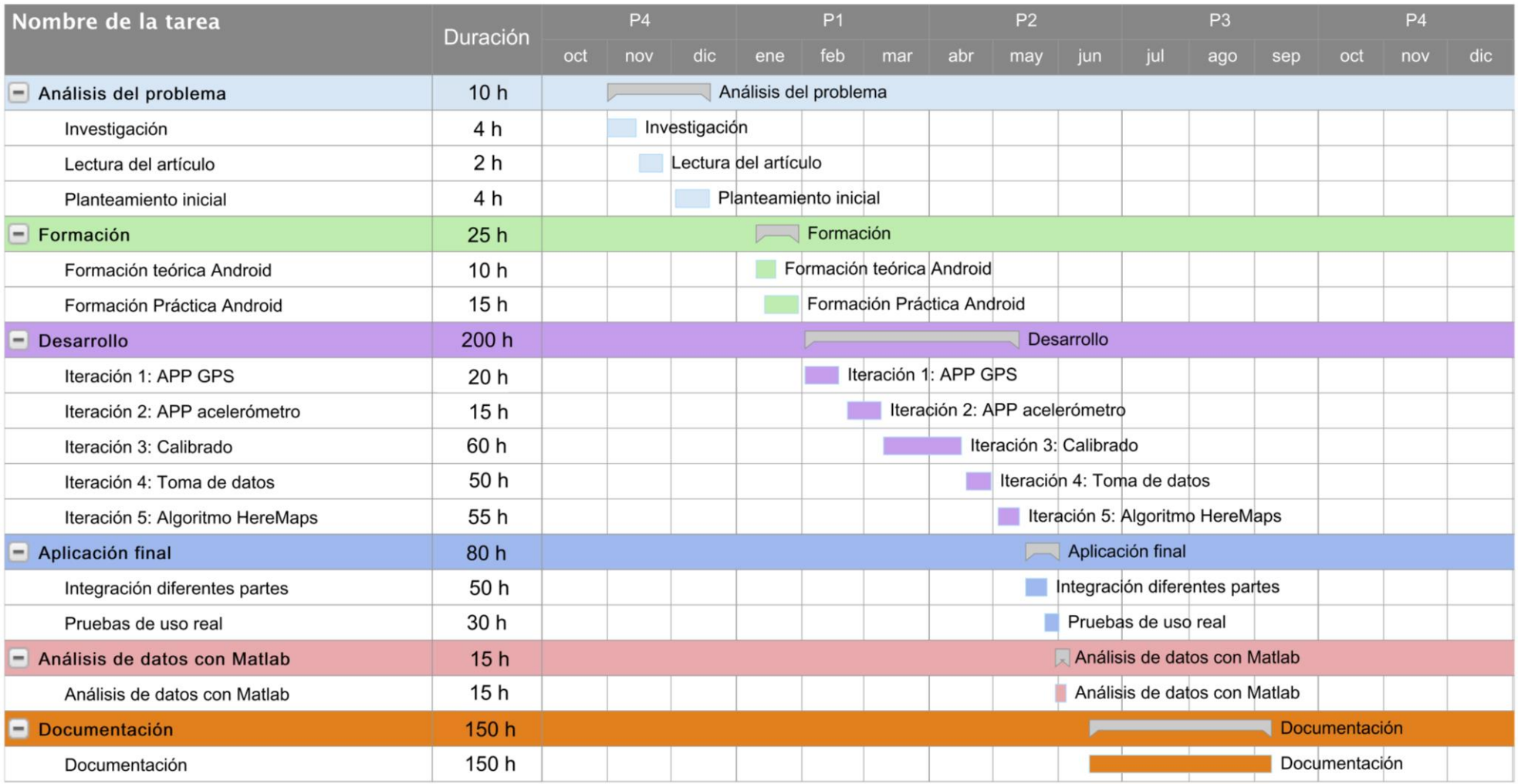

Figura A.2: Diagrama de Gantt

PERT

Para un mejor análisis y comprensión de la planificación y dependencias de las tareas, se definirá un diagrama de PERT. PERT es un método para analizar las tareas involucradas en completar un proyecto dado. Para ello definiremos una tabla de dependencias:

Tarea	Letra asignada	Dependencias
Investigación	A	-
Lectura del artículo	B	-
Planteamiento inicial	C	A, B
Formación teórica Android	D	A
Formación práctica Android	E	C
Iteración 1: APP GPS	F	D, E
Iteración 2: APP acelerómetro	G	D, E
Iteración 3: calibrado	H	F, G
Iteración 4: Toma de datos	I	H
Iteración 5: Algoritmo Here WeGo	J	I
Integración de diferentes partes	K	J
Pruebas de uso real	L	K
Análisis con Matlab	M	K
Documentación	N	L, M

Tabla A-4: Tareas y su correspondiente dependencia.

Con la realización de este tipo de gráfico, podemos aproximar la envergadura del proyecto, de haber sido realizado de continuo y con un mayor número de recursos.

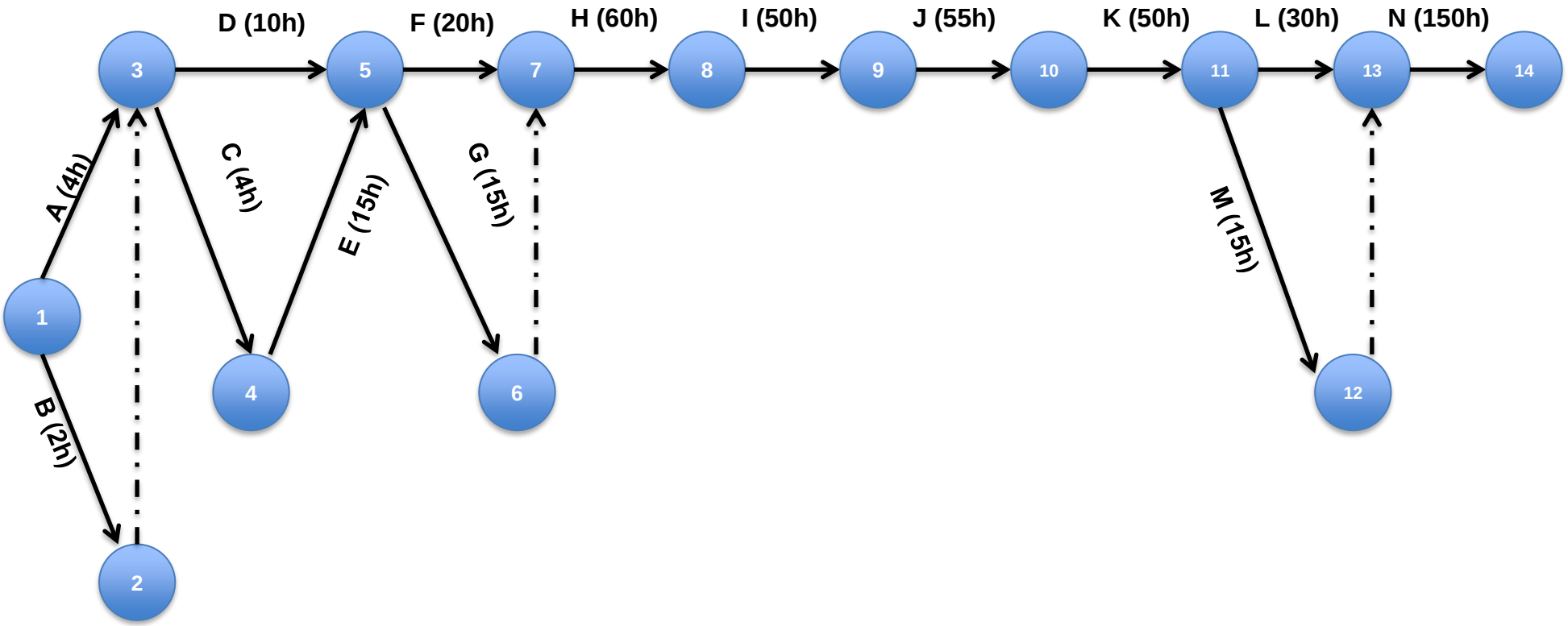

Figura A.3: Diagrama de PERT.

Presupuesto

A lo largo de este apartado se detallará el presupuesto final del proyecto. Para ello se tendrán en cuenta los gastos de personal, hardware, software y otros gastos necesarios para la realización del proyecto.

Comenzamos el presupuesto por el coste de personal. Se ha tenido en cuenta para ello el salario medio de un Graduado en Ingeniería con una experiencia menor de 2 años. Se estima en 27000 € anuales de forma aproximada. Además, debido a la colaboración y consulta para la orientación durante el proyecto, de un ingeniero superior, se necesitará el salario medio de un Ingeniero superior Sénior, que estimamos en unos 45000 € anuales. En ambos casos son salarios brutos, teniendo en cuenta que el número de horas de trabajo anuales asciende a 1680, el coste de cada tipo de ingeniero por horas será:

- Graduado en ingeniería (menos de 2 años de experiencia): 16,07 €/h.
- Ingeniero superior (sénior): 26,78 €/h.

Para el cálculo de horas trabajadas, podemos tomar el número de horas de la planificación final, 480h, en el caso del graduado en ingeniería. Además, para el Ingeniero superior, podemos estimar un número de horas de aproximadamente 60, entre reuniones y demás colaboraciones. El presupuesto para esta parte lo podemos ver en la siguiente tabla:

Personal	Cargo	Salario	Tiempo	Coste [€]
Álvaro González Caballero	Graduado en ingeniería	16,07 €/h	480 h	7713,60 €
Celeste Campo Vázquez	Ingeniera superior	26,78 €/h	60 h	1606,80 €
TOTAL				**9320,40 €**

Tabla A-5: Coste del personal.

Cabe destacar, que a pesar de que algunas tareas (señaladas en la tabla A-2, han sido elaboradas de forma conjunta con otro graduado en ingeniería (equivalente a 130 horas de las 480 horas totales del proyecto), se ha decidido no tenerlo en cuenta en los presupuestos del proyecto. El fundamento se basa en que dichas horas serán contabilizadas en su propio proyecto. A efectos de justificación de horas, si se tratase de un proyecto de empresa, estas 130 horas podrían corresponder a un curso de formación sobre Android, y la reorientación del acelerómetro, que a cada graduado pagaría su propio proyecto.

Continuamos el presupuesto, con la suma de los costes de infraestructura y material. A nivel Hardware se ha necesitado el uso de varios dispositivos móviles y un ordenador portátil. Suponiendo una vida útil de 3 años para los dispositivos móviles, y de 5 años para el portátil, podremos calcular su amortiguación. Esto nos permitirá ser más exactos a la hora de imputar costes a nuestro proyecto.

Podemos calcular los costes imputables a partir de la fórmula $\frac{A}{B} \times C \times D$, donde:

- A = Número de meses de uso del equipo.
- B = Vida útil.
- C = Coste total de equipo.
- D = Porcentaje de uso del equipo dentro del proyecto (en este proyecto será siempre el 100%).

Equipo	Coste (€)	% Uso dedicado al proyecto	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable (€)
Smartphone BQ Aquaris 4.5	178,04	100	10	36	49,45
Smartphone Samsung A5	167,66	100	10	36	46,57
Ultrabook Macbook Pro Retina 13’	1550,01	100	10	60	258,33
TOTAL					**354,35 €**

Tabla A-6: Costes infraestructura. Hardware.

Por otro lado debemos considerar los costes en software. Por tratarse de un proyecto de tipo académico, hemos podido utilizar el software gratuito que tiene la Universidad Carlos III a disposición de sus alumnos. A pesar de ello algunos programas han necesitado de la adquisición de licencia.

Software	Coste (€)	% Uso dedicado al proyecto	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable (€)
Licencia de OS X	0	100	10	-	0
Licencia Microsoft Office	0	100	10	-	0
Matlab	500,00	100	10	-	500,00
Here WeGo	0	100	10	-	0
Android	0	100	10	-	-
TOTAL					**500,00**

Tabla A-7: Costes infraestructura. Software.

Por último, aunque el proyecto se ha realizado en un espacio dedicado del hogar, o en la universidad, se ha necesitado de otro tipo de infraestructura como: una tarifa de datos, un coche de pruebas y combustible, que son gastos imputables al proyecto. Puesto que es difícil de cuantificar todo lo relacionado con el vehículo, ya que no sólo hay que cubrir los costes del mismo, sino los derivados de su utilización tales como: seguro, desgaste, impuestos, etc. Se ha decidido (basándose en estudios de la DGT), el establecimiento de un precio de 0,40 € por kilómetro de media para los 4 vehículos distintos utilizados. En total se han realizado unos 1500 kilómetros probando la aplicación, por lo que la cifra asciende a 600€.

Equipo	Coste	% Uso dedicado al proyecto	Dedicación	Periodo de depreciación (meses)	Coste imputable (€)
Vehículos (detalle en 1.5)	0,40 €/km	100	1500 (km)	-	600,00
Tarifa de datos + internet (Vodafone ONE M)	48,80 €/mes	100	10 (meses)	-	488,00
TOTAL					**1088,00**

Tabla A-8: Otros costes.

Conocidos todos los costes por bloques, podemos proceder al cálculo de los costes totales. El coste total del proyecto asciende a once mil doscientos sesenta y dos euros con ochenta y cinco céntimos de euro.

Tipo de coste	Cantidad (€)
Costes de personal	9320,40
Costes de infraestructura (hardware)	354,45
Costes de infraestructura (software)	500,00
Otros costes	1088,00
TOTAL	**11262,85**

Tabla A-9: Costes totales.

Por último, se recuerda que esto es un proyecto de desarrollo, por lo que en ningún momento se ha tenido en cuenta para la gestión y presupuestado las tareas de mantenimiento. Se considera que dichas tareas corresponden a una actuación posterior, iniciada al fin del presente proyecto.

Anexo B: Manual de usuario

Este anexo es una breve guía de usuario. Está destinado a explicar el funcionamiento general del sistema.

Al iniciar la aplicación, nos encontramos con una pantalla que muestra un mapa de la ubicación donde nos encontramos y un punto azul. Este punto azul muestra la ubicación del usuario en tiempo real.

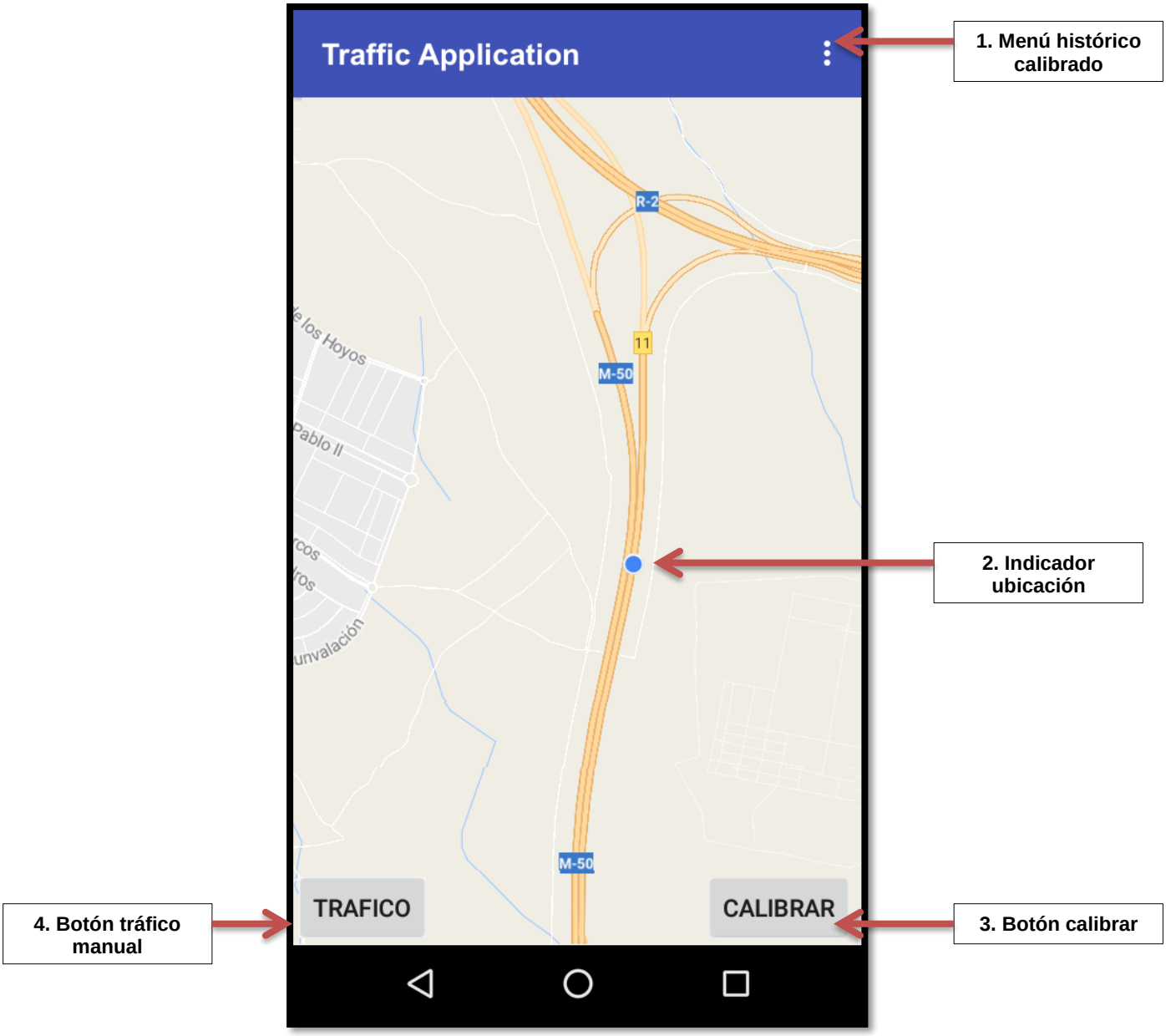

Figura B.1: Vista principal de la aplicación.

Encontramos los siguientes controles accesibles para el usuario:

- **1. Menú histórico calibrado:** Permite desplegar un menú para la cargar/guardado de una calibración.
- **2. Indicador ubicación:** Indica la ubicación actual del usuario.
- **3. Botón calibrar:** Inicia una nueva calibración, y permite avanzar en las diferentes fases de la misma.
- **4. Botón tráfico manual:** Permite almacenar un evento manual de tráfico.

Para comenzar a usar la aplicación, es necesario realizar un primer calibrado. A continuación, explicamos el método.

Fase de calibración

Esta fase da comienzo después de ser pulsado el botón calibrar visto en la figura B-1. La calibración del acelerómetro para ser reorientado consta de dos fases: Fase de reposo y Fase de movimiento.

Fase de reposo:

Da comienzo tras ser pulsado por primera vez el botón calibrar. Se mostrará por pantalla el mensaje: “Calibrado en curso, mantenga el vehículo en reposo”. Este mensaje durará en pantalla aproximadamente 5 segundos. El usuario deberá permanecer en reposo junto con su vehículo.

Figura B.2: Detalle del mensaje calibrado. Fase de reposo.

Una vez terminada la fase de calibración en reposo, la aplicación avisará al usuario y dará comienzo la fase de movimiento.

Fase de movimiento:

El usuario podrá leer en pantalla el mensaje: “Calibrado en curso. Pulse ‘CALIBRAR’ y comience a conducir”. Una vez el usuario haya pulsado el botón calibrar, se dará inicio a la fase de calibrado en movimiento. En esta fase el usuario únicamente tendrá que conducir. La aplicación por si misma tomará los datos necesarios para la calibración del sistema.

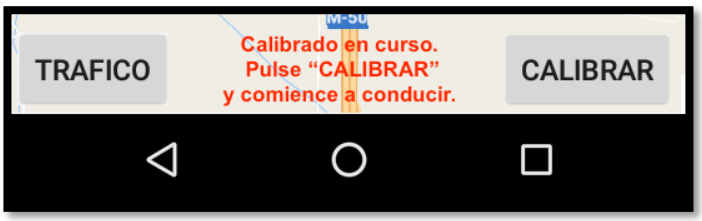

Figura B.3: Detalle del mensaje calibrado. Fase de movimiento.

Una vez acabada la fase de calibración, el usuario recibirá una notificación en forma de mensaje. Este texto contendrá: "Calibración realizada con éxito" y vendrá acompañada de la aparición de nuevos elementos en la interfaz de usuario que explicaremos más adelante.

Toma de datos

Esta fase comenzará de forma automática tras la finalización de la fase de calibración. En esta fase la aplicación comenzará a detectar eventos de congestión en el tráfico de forma automática. Contaremos con la interfaz de la siguiente figura. Los nuevos elementos aparecidos son:

- **1. Indicador valores aceleración vehículo:** Indican el valor de aceleración para los ejes a_X, a_Y y a_Z del vehículo.
- **2. Indicadores aceleración/frenado:** Indican, una vez que la aplicación ha sido calibrada correctamente, si el coche está acelerando o frenando en tiempo real. Para ello se utilizará una letra 'A' en color verde para la aceleración y una letra 'F' en color amarillo para el frenado. De encontrarse ambas inactivas figurará en ambos colores el signo '-'. (Este indicador es de utilidad para saber el correcto calibrado. Si en estado de reposo el valor de Z es de al menos 9.8, es una buena señal).

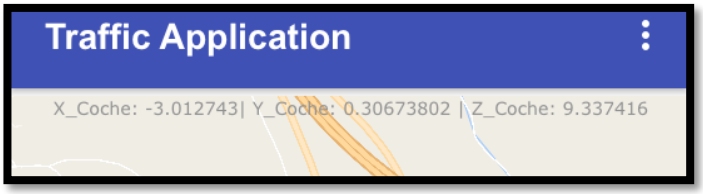

Figura B.4: Detalle del indicador de valores para la aceleración del vehículo.

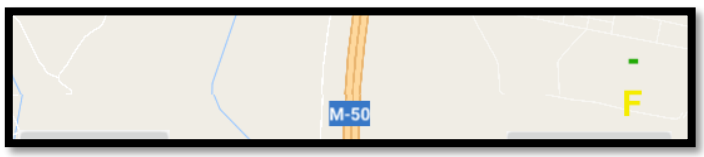

Figura B.5: Detalle de los indicadores de aceleración/frenado.

Figura B.6: **Vista de la aplicación calibrada.**

Datos manuales

Existe una opción que posibilita al usuario la introducción de eventos de congestión detectados de forma manual. Cuando estos eventos sean introducidos, aparecerá en la interfaz un indicador de color azul en la posición donde el usuario se encontraba.

Datos automáticos

La aplicación detecta por sí sola aquellos eventos de congestión de tráfico automática. Cuando estos eventos sean introducidos automáticamente, aparecerá un indicador de color rojo en la posición donde se detectó.

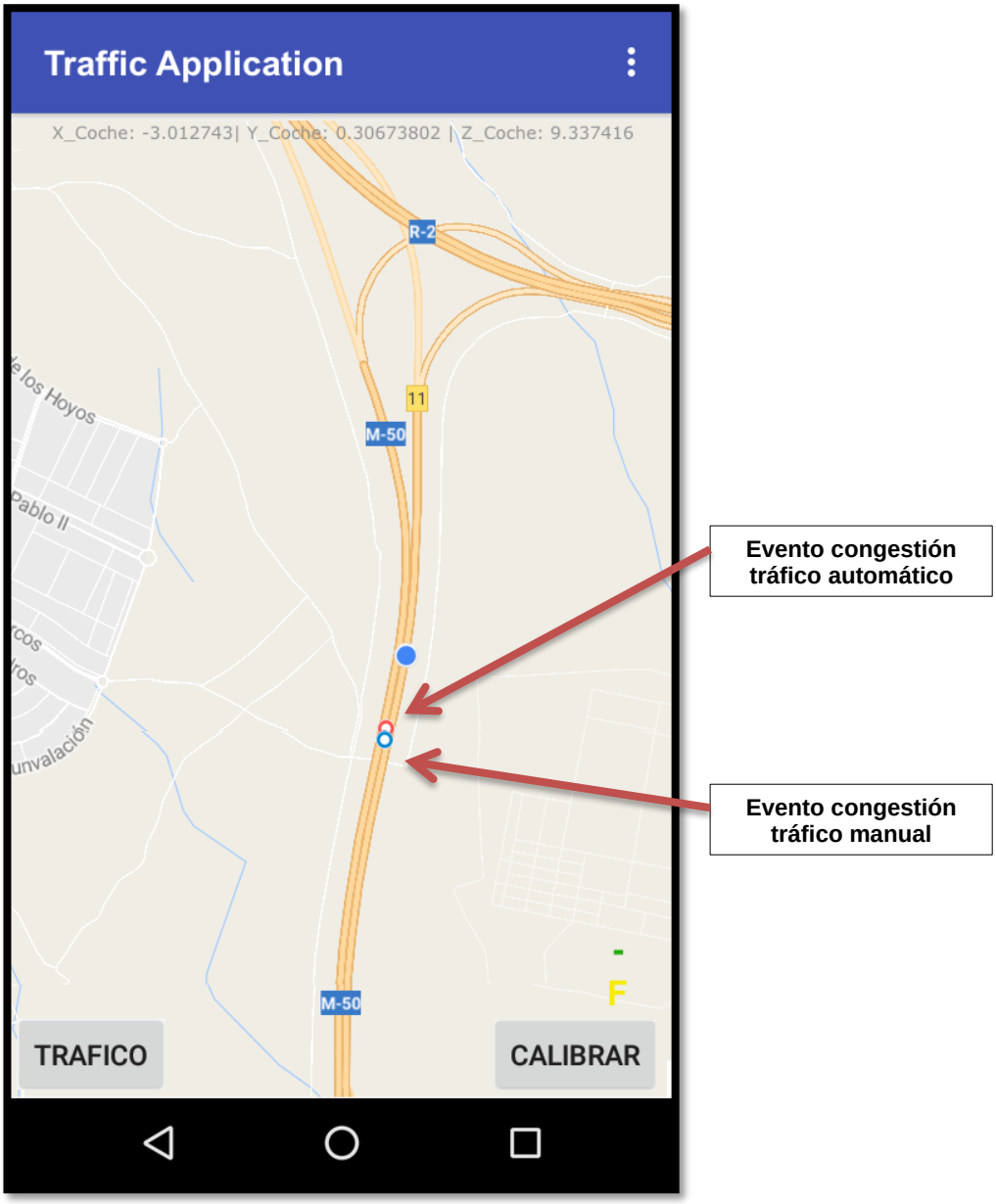

Figura B.7: Vista de los eventos manual y automático detectados.

Cargar/Guardar Calibrado

Con el fin de facilitar el manejo de la aplicación, se podrá guardar, con el objetivo de ser guardado posteriormente, un calibrado. Esto será de gran utilidad para vehículos donde se utilicen soportes, siendo la orientación del dispositivo siempre la misma.

Algunas consideraciones

- Para el correcto funcionamiento de la aplicación se deberán dar los permisos requeridos de sistema.
- Se deberá cumplir las leyes locales de privacidad.
- Se deberá cumplir las leyes locales pertenecientes a la circulación y manejo de vehículos.

Annex C: English version

1. Introduction and goals

In this first chapter, we will make an introduction to the topic: Traffic, and how to monitor road traffic conditions using new technologies. First of all, we will explain the different goals to get, then, the project management phases, and last of all, the main structure of this document.

1.1. Introduction

The shifting of people and goods is a fact linked with the human being. Since the beginning, traffic is a historic, social, economic and legal phenomenon that concerns to the biggest part of societies. The cause is the vehicle flow that travels across streets or roads. [1]

There are some different ways to study traffic. Among other things, there are studies about traffic accidents, traffic lights, an intersection of avenues, etc. Although the case we are going to study along this Bachelor Thesis, is the congestion. [2]

Illustration 1: Congestion on Madrid highway A4.

Congestion[3] (also known as traffic or traffic jam) happens due to saturation on roads. If the quantity of vehicles for a road, is higher than the rate that road can absorb, traffic will appear. This usually happens during rush hours, and it is the main reason for drivers to waste their time and patience. It also causes economic loss (vehicles in these conditions waste their fuel), a reduction in productivity and an increment on pollution.

Ranking	Urban Areas	Total loss hours on 2014
1	Barcelona	25
2	Madrid	22
3	Sevilla	18
4	Bilbao	16
5	Zaragoza	12
6	Valencia	11

Table 1-1: Ranking of loss hours in the most congested urban areas of Spain. The year 2014. (Source INRIX)

Due to the globalisation of traffic problem, during the history, and thank the human evolution, people has developed different solutions to this problem. Some of these solutions are linked to the development of new technologies. It is possible to create electronic devices, light in weight, and easy to use and transport. The best example is the mobile devices. This kind of gadget has features as powerful processors, or different sensors (accelerometer, GPS, different kind of connectivity).

This features, and the easy to use the devices, help mobile devices to be one of the most useful gadgets for drivers. Thanks to the different sensors on the devices, it is possible to communicate with other users, and share information.

Now, I will try to resume some solutions to the congestion problem. These solutions are based on different points of view. Some of them use mobile devices to solve the problem:

1.1.1. Technological solutions

Nowadays, there are some technological solutions with the aim of detecting and prevent congestion situations. Here some examples of this type of technologies:

Smartphone Applications: This APP's let users share traffic information on real time. There are different types of applications: coopering, based on algorithms, base on data. This type of solution is going to be explaining with more detail in the next chapters.

Autonomous technology: An autonomous vehicle is able to sense the environment by itself. It can take its own decisions, based on the information its sensors capture. This technology is connecting in real

time to the network, which provides the different type of information to choose its own route or the suitable speed during the travel.

Radio Data System (RDS): This technology let the addition of information to radio programs, in FM radio band. One of the most useful applications is as a receptor of traffic information. It let automatically to change the current broadcasting, to one that is transmitting traffic information in that moment. For the correct work of this feature, the radio system must be turn on.

1.1.2. Urban solutions

Congestion usually appears on big cities road access. Some objectives facts build this problem:

- 72% of European citizens live in urban places.
- Cities are the place which consume 75% of energy, and which create 85% of wealth on the European Union.
- 98% of journeys correspond to urban one, less than 50 kilometres of distance.

European cities such as Barcelona, London, or Madrid, are developing different solutions for this problem. For instance, Madrid develops the "Sustainable Urban Mobility Plan for Madrid" (SUMP) [4].

The SUMP is defined on the European "Developing and implementing a Sustainable urban mobility plan", as a strategy plan design to solve people and companies mobility needs in cities. The aim is to improve life quality.

On the SUMP of December 2014, Madrid shows different solutions develop since 2006. This plan makes a diagnosis of the situations, to define different goals and steps, to manage the urban mobility in a sustainable way.

To get these road security goals (reduce the number of accidents, and reduce the number of accidents victims), the SUMP defines the next steps:

- To promote pedestrian mobility.
- To promote public transport.
- To promote cyclist mobility.
- To improve motorbike mobility.
- To improve the taxi service.
- To use share mobility methods.
- To improve the accessibility on public transport.

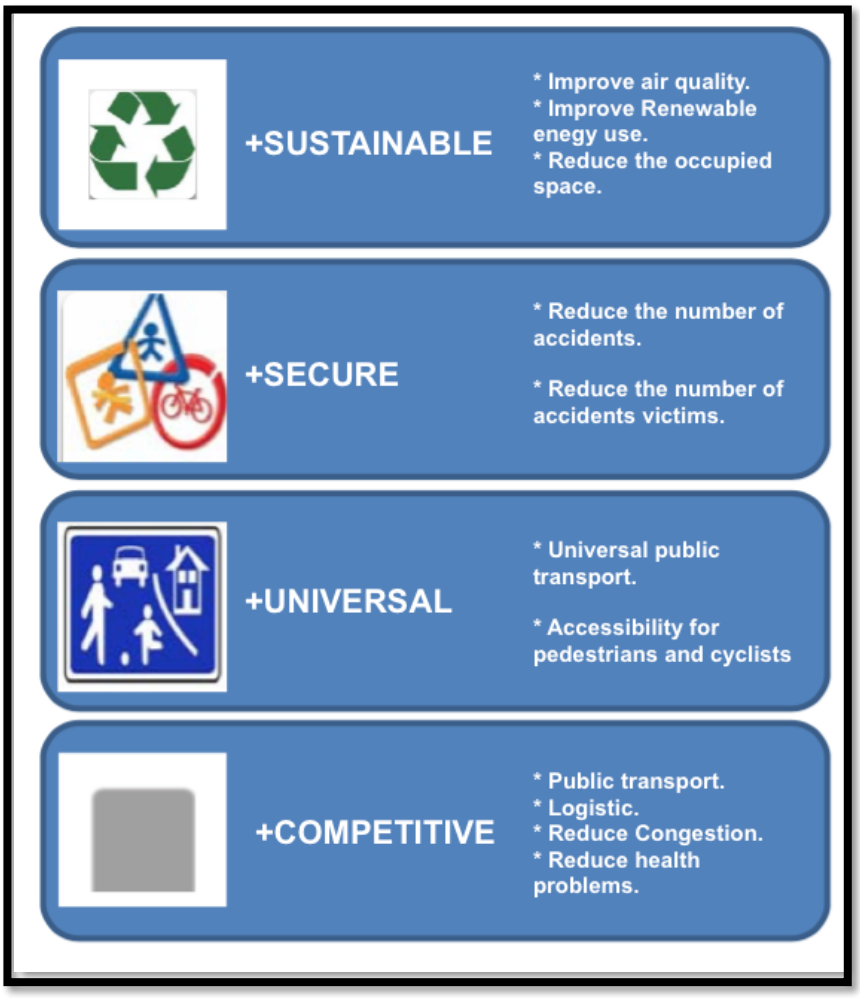

Illustration 2: Madrid SUMP (December 2014). (Source translation of PMUS Madrid)

1.1.3. Others solutions

There are also other types of solutions that help to solve congestion problems, some of them on an indirect way. Here there are some of them:

Dirección General de Tráfico (DGT): This is a Spanish governmental organisation to manage traffic. It was created in 1959 to focus on the high number of vehicles problem. Nowadays, it is the responsible for security on the roads. It promotes consciousness-raising, offers traffic prediction, and offers other traffic information too.

Public transport: This is one of the issues that governments try to resolve. Cities, national governments and private companies too, fight for an increase in the use of a public method of transport. This could help to solve traffic problems.

1.2. Objectives

This Bachelor Thesis looks for a solution to get traffic information with an Android application. The idea is to develop an alternative algorithm, different than the one that APP's use today. The biggest part of these applications are based on an algorithm, that sends information to an external server of their companies. This project looks for an alternative solution, that uses the information in a local way. For that purpose, a smartphone and its sensor will became especially helpful, to detect congestion events.

Before starting developing the application, and based on the different requirement, we define some objectives.

1.2.1. Project objectives

The main objective is developing an Android application base on an algorithm that detects traffic in a secure way. It uses the different capabilities of a Smartphone, as sensors, or processor power. It is important to preserve user's privacy.

If we have a look at the most common solutions for this problem, we will see that the biggest part of them send user's data to servers. In this situation, the user loses data control. The privacy is in danger, and the resources as battery life, or internet data traffic, are wasted. We set the next secondary objectives to solve these situations:

- The develop will respect user's privacy. To get this goal, the algorithm will detect the traffic events. Then, only events information will be sent to the network. This will avoid that anyone knows the users' route of navigation. The event detection will be local, into the device, and not on external servers.
- Resources will be used in an effective way. This mean that we take care of the data and battery consume.
- The most important part of this project is not the application. Instead of that, we emphasise the importance of the developing an alternative algorithm.

1.2.2. Personal objectives

The election to develop Android software is based on the next personal objectives:

- Learn how to develop and implement mobile applications, on an Operative System as Android, has a lot of advantages.

- Study how to use agile development methodologies, as Scrum. This kind of methodologies are very popular today, to develop projects in a properly way.

1.3. Development stages

The development of this project was in different stages. In this section, we will describe then on a rising order.

Stage 1 – Analysis of the problem: During this first stage, we analyse the problem and then, we develop alternative solutions. We will keep in mind our objectives. This is a documentation stage, that will let us deal with the next stages in a successfully way.
Immediately after there are some of the articles we had a look during the analysis of the problem stage:

- P. Mohan, V.N. Padmanabhan and R. Ramjee. "TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones", *Technical Report MSR-TR-2008-59*, Bangalore, India, 2008. [27]
- S. Ayub, A. Bahraminisaab y B. Honary. "A sensor Fusion Method for Smartphone Orientation Stimation", presentada en 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, Junio 2012. [31]
- R. Bhoraskar, "Traffic and Road Condition Estimation using Smartphone Sensors.", Bachelor Thesis, Department of Computer Science and Engineering, Indian Institue of Technology, Bombay, Mumbai, India, 2011. [30]

Stage 2 – Training: This stage is dedicated to learning all the skills necessary to develop the project. Android studio is the Integrated Development Environment (IDE) official for Android. To develop in this platform is necessary knowledge on XML and Java programming languages.
Some of the resources use during Android training are:

- C. Campo Vázquez y C. García Rubio, Curso de Android de la asignatura Aplicaciones móviles, Universidad Carlos III de Madrid, Leganés, 2015. [32]
- "Web para desarrolladores de Android", *Google*. [17]
- S. Gómez Oliver, *Curso Programación Android*, 3th ed. Publicación libre, 2015. [33]

Stage 3 – Development: Using the tools we learn on stage 2, in this new stage we will start developing. First of all, we will create some applications using the different sensors. These applications will be tiny modules of the final application.

In this Stage, we will also use Scrum methodology. The develop will take different iterations. This will help us to develop the software by the time we learn how to use Android, and the different Smartphone sensors. After this stage, everything is prepared to develop the final application.

Stage 4 – Final application: During this stage, we develop the final applications. It is very important to take care of all the established objectives. We will integrate the different modules developed during Stage 3.

Stage 5 – Testing the application, and see the results: This stage is one of the most important. We will test all the different parts of the application. This stage also is useful to establish the final configuration of the different sensors. After this stage, the application must work on a properly way.

Stage 6 – Documentation: This is the final stage. It is created to document the project. We will redact the current report.

1.4. Resources

In this section, we will describe the used resources to develop this project. They are classified on Hardware, Software, and Others.

Hardware:

- Smartphone Android, BQ brand and Aquaris 4.5 model.
- Smartphone Android, Samsung brand and A5 model.
  Both Smartphone are used to make the test of the application, in a real environment. The election of then is based on their technical characteristics.
- Ultrabook Apple brand and MacBook Pro retina 13' model. It is used to develop the application, to analyse the different data, and finally to write this report.

Software:

- Mac OS X 'El Capitan' 10.11.4. It is the default operative system for the ultrabook.
- Android Marshmallow 6.0. It is the default operative system for both Smartphone.
- Android Studio (IDE). It is the IDE provided by Google.
- Microsoft Office 2011. It was used to redact this report. We used Word, and Excel Microsoft tools.
- HereMaps API. It is used to take the maximum speed on roads.

- Matlab R2015b. We chose this software because probably is the most powerful in its type in the market.

Other:

- Vehicle: Peugeot 406.
- Vehicle: Opel Vectra.
- Vehicle: Audi Q5.
- Vehicle: Citroën C8.

The use of different vehicles is based on the idea of taking the biggest part of the available vehicles type for the public in general.

1.5. Document structure

This section will make a general description of document structure in general.

1) **Introduction and objectives:** In this first chapter, we will introduce the topic: The traffic, and how to monitor road traffic conditions using new technologies. First of all, we will explain the different goals to get, then, the project management phases, and last of all, the main structure of this document.
2) **State of the art:** In this chapter, we will revise the specifications of this bachelor thesis. For this task, we will analyse the different solutions for the traffic problem that we can find currently in the market. We will have a look at different technologies as for example, the development of mobile gadgets, the sensors, or the mobile operative systems.
3) **The design of the technique solution:** In this chapter, we will describe how we solute the problem. We will describe the architecture, and then we will show the requirements and the use cases.
4) **System implementation:** In this chapter, we will describe how to develop the technique solution.
5) **Assessments and results:** In this chapter, we will test the developed solution. We will test the different operations, and we will analyse the results to determine if it is a correct solution.
6) **Regulatory framework:** In this chapter, we will have a look at the Europe and Spanish legislation that affects to this project.
7) **Conclusion:** This is the last chapter. We will show our conclusions. We will revise the different objectives, and work lines.

2. State of the art

In this chapter, we will revise the specifications of this bachelor thesis. For this task, we will analyse the different solutions for the traffic problem that we can find currently in the market. We will have a look at different technologies as for example, the development of mobile gadgets, the sensors, or the mobile operative systems.

2.1. Contemporary solutions

These days, there are different solutions to provide traffic information on real time with applications. Each one uses a different way to detect this kind of information. Here there is some information about these applications:

2.1.1. Google Maps

Google Maps is probably the most popular navigator. It offers satellite imagery, street maps, real-time traffic conditions, route planning, etc. We will focus on their traffic conditions algorithm. Google uses mainly three sources of information:

Sensors: Since the beginning of Google Maps, they use road and vehicles sensors to monitor traffic.

Database: It is one of the most important sources of information. They use historical traffic data to predict future traffic conditions. Nowadays Google continues using this kind of detection, and it is available on their web.

Users: In 2007, Google created an algorithm to detect traffic congestion in real time. It uses user's information. If users enable Google Maps, and the use of GPS, users phone sends bits of data back to Google. This data describe how fast user is moving. With thousands of users' data, Google servers are able to detect when there is congestion on a road or street.

2.1.2. Dirección General de Tráfico

DGT is the most important traffic public organism of Spain. Their main objectives are to promote security on streets and roads. This organism has some tools to detect traffic state.

They use their web and application to offer traffic information. DGT has tens of cameras and sensors in different roads of Spain. They also provide another type of information as the weather forecast.

2.1.3. Waze

There are other types of traffic detection that need user collaboration. One of the best examples is the Waze navigator. It provides an interface, to let users inform about roads conditions. The collected information will be immediately available for other users. The name for this type of applications is collaborative.

2.2. Mobile devices

Since 2007, the mobile market has improved its technology incredibly fast. Nowadays there is a large list of different mobile devices, as for example smartphones, tablets, smartwatches, etc.

All of then provide a big computational power and a lot of sensors. These sensors can keep mobile devices with extra capabilities as GPS, accelerometer, mobile Internet.

2.3. Software development for mobile devices

Linked to mobile devices market development, it has appeared another important market. It is the Software development for mobile devices. There are two players in this market that are outlined. They are Android of Google and iOS of Apple.

Although both of them have some advantages we have chosen Android to develop our application. Here there are some of the reasons:

Android is a mobile operating system developed by Google, based on Linux kernel. It has the largest installed base of all operating systems of any kind. This is because Google releases Android's source code under open source licenses.

3. Design of the technique solution

In this chapter, we will describe how we solve the problem. We will define the architecture, and then we will show the requirements and the use cases.

3.1. Architecture

To have a better understanding of the application architecture, we have developed a layer division.

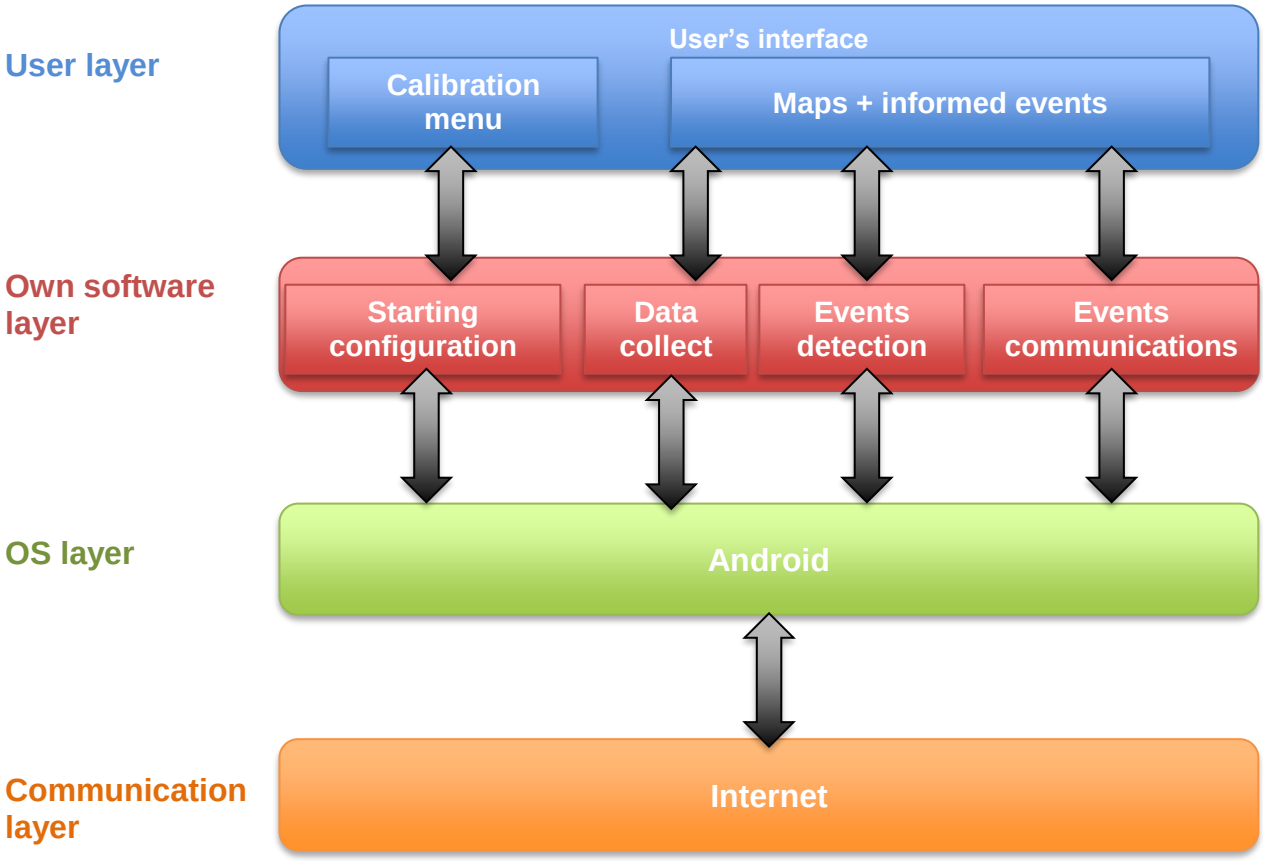

Illustration 3: Application layer architecture.

This layer architecture makes possible intercommunication between each layer. This is very helpful because each layer can use services of their behind layer.

3.2. Requirements

A software specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements. For this bachelor thesis, we have made a classification in functional requirements, non-functional requirements, and restriction requirements. Now, there is a summary of all of them.

3.2.1. Functional requirements

Functional requirements may be calculations, technical details, data manipulation, and procession, etc. We have defined the next functional requirements:

- Sensors calibration: An option to calibrate sensor must exist.
- Calibrations verification: The system will provide a way to inform when the calibration process is finished.
- Calibration data: The system will show accelerometer data on the screen.

- Traffic congestion event detector.
- Message to inform about a traffic event.
- A button to inform manually about traffic event must be available.
- Log of traffic events.

3.2.2. Restriction requirements

Restriction requirements may be those that must be a compliment. They must work for the properly operation of the system.

- The user must accept permissions to let the applications use the sensors.
- Devices that run this application must have GPS an accelerometer.
- Devices that run this application must have Internet access.

3.2.3. Non-functional requirements

Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviours.

- The application will provide a way to save the actual calibration.
- The application will provide an option to load a previous calibration**.**
- The application will handle user privacy in a properly way.

3.3. Use cases

In software and systems engineering, a use case is a list of actions or event steps, typically defining the interactions between a role. To sum up this section, we will make a use cases diagram of the application.

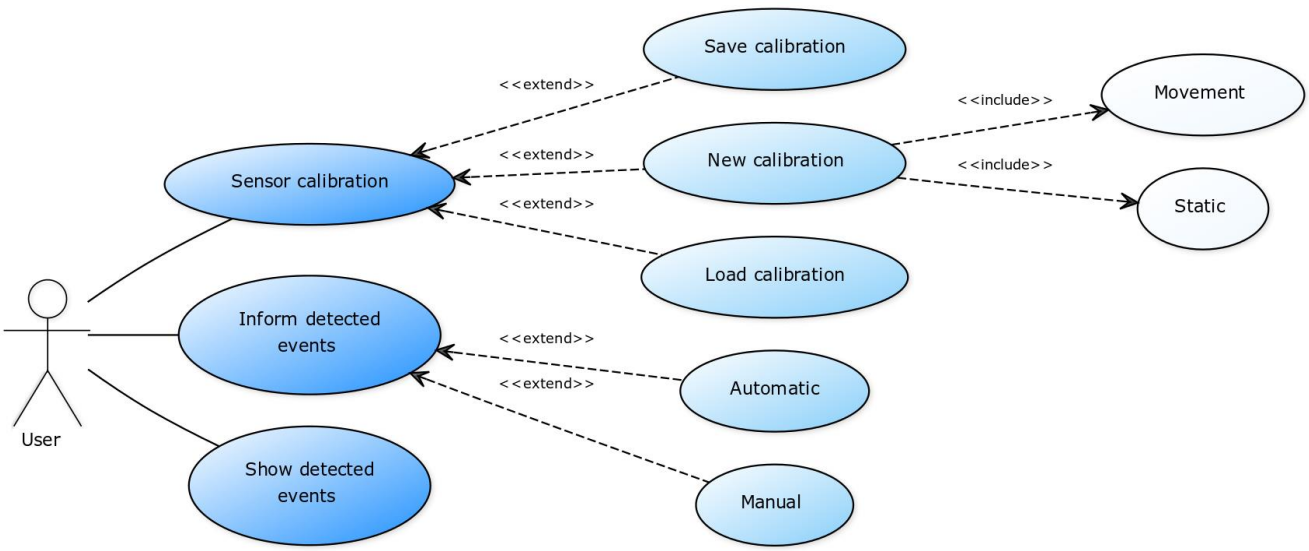

Illustration 4: User use diagram.

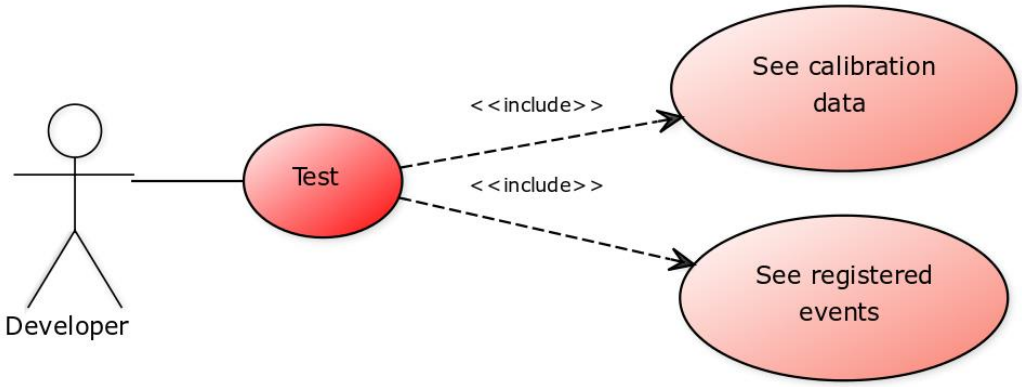

Illustration 5: Developer use diagram.

4. System implementation

In this chapter, we will describe how to develop the technique solution. For this part of the project, it will be very important to acquire experience in Android language.

4.1. Agile Software Development

During the development of this project, we have used the agile software development methodology. This kind of methodology is very helpful in investigation projects. It promotes adaptive planning, evolutionary development, early delivery, and continues improvement. It also encourages rapid and flexible response to change. Here there is a diagram of Scrum methodology:

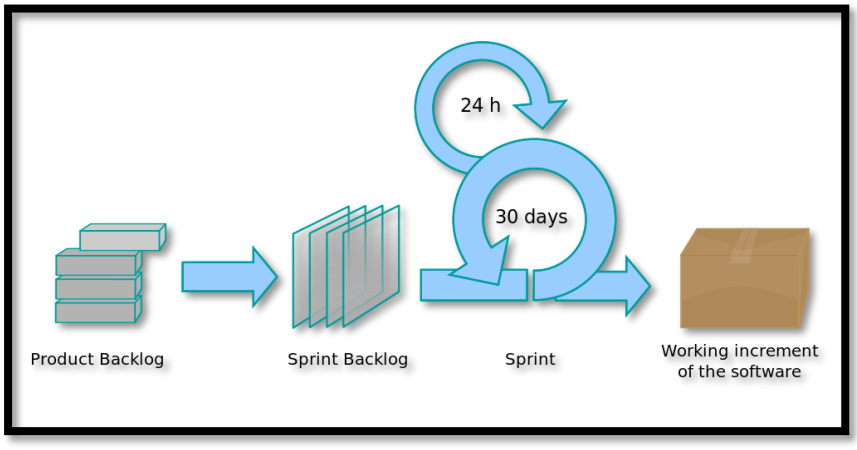

Illustration 6: Scrum methodology diagram. (Source Wikipedia)

We have chosen this type of methodology because it helps to learn the use of Android, while we start to make the different modules of the applications.

4.1.1. GPS Application

As the first contact with Android sensors, this application is a good introduction to Android development language. We will need to our last application different information as user location, user speed, etc. The GPS sensor will provide this information.

4.1.2. Accelerometer Application

It is the second iteration of our development. Android, as with GPS sensor, has some classes that help with the development of the code. During the implementation of this application is very important to know which is the coordinate system defined.

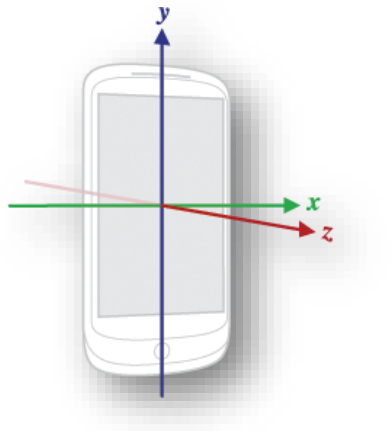

Illustration 7: The coordinate system of an Android device. (Source Android developers Web)

This image shows the axis orientation. Accelerometer information will be shown on the metric unity m/s^2.

4.1.3. First approximation: Calibration

We have implemented a solution based on the article: *TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones* [27]. In general, the phone axes could be in an arbitrary orientation with respect to the vehicle axes. The idea of this bachelor thesis is to detect a period of braking that could be a traffic event. For that proposal it is very important to reorient the accelerometer axis, to have the same orientation as vehicle ones.

The use of Euler Angles simplifies our calculations significantly. For more information, see the article. Once our accelerometer is reoriented, we are able to detect when the vehicle is braking.

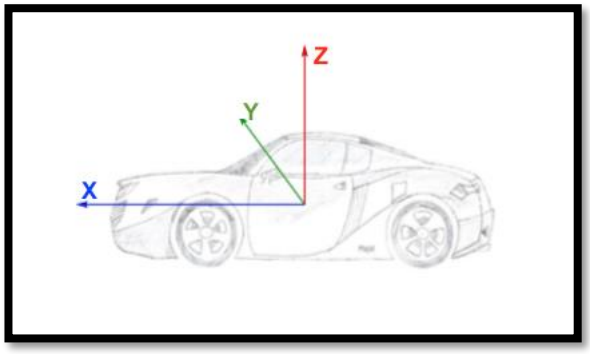

Illustration 8: The coordinate system of a vehicle.

4.1.4. Second approximation: Taking data

Once the calibration application is working, we can start taking data. It is important to establish a correct detection threshold. For that reason, we created an application that takes braking events data. We also implement a button to introduce manually traffic events detected by the user.

After some sessions probing the application, we took enough data to represent it on a graphic. The next graphic shows the instantaneous speed, and manual traffic was taken. We can conclude with these data, that it is not enough to detect traffic events, so we will make a new iteration to implement a new solution.

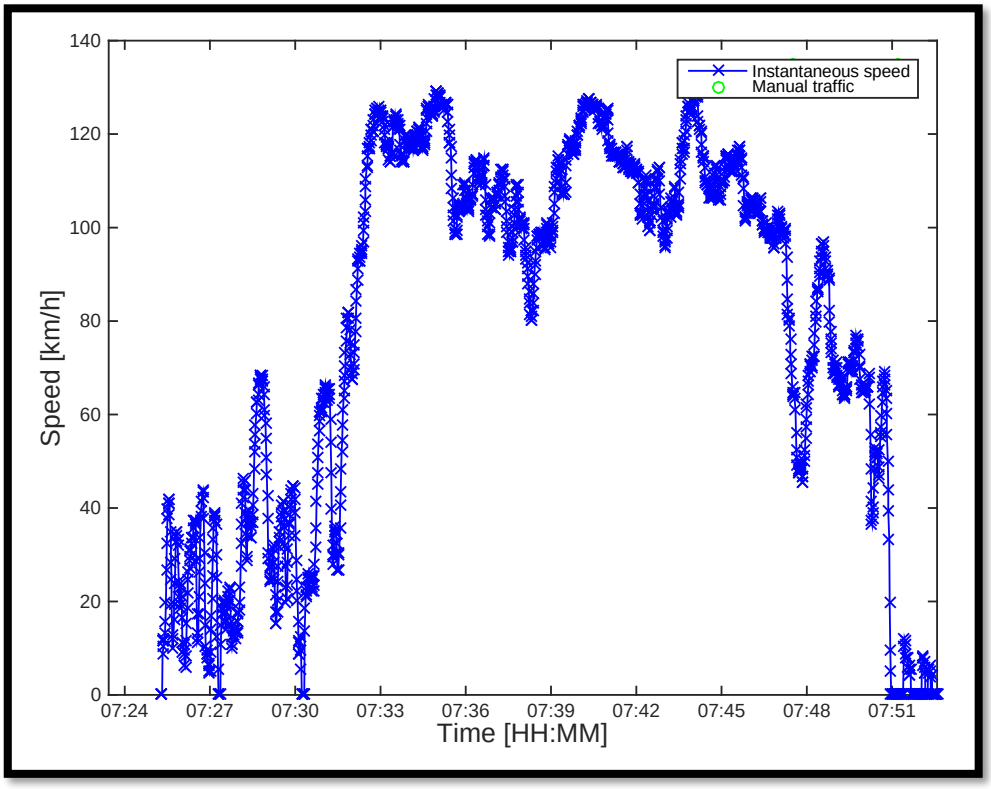

Illustration 9: Instantaneous speed and manual traffic events.

4.1.5. Third approximation: Verification

This iteration is the last one. We implemented a new algorithm to detect traffic automatically. It is based on users speed. Thanks to the HereMaps API we have access to speed limit information for the different roads. The idea is the next:

1) The application detects a braking (with the accelerometer reoriented).
2) The application takes speed limit of the road.
3) The application takes the instantaneous speed of the vehicle.
4) The application compares both data and decides if the user is on a congestion event.

The result of this solution is in the next graphic. As we can see this new detection algorithm is working. The automatic events coincide with the manual events:

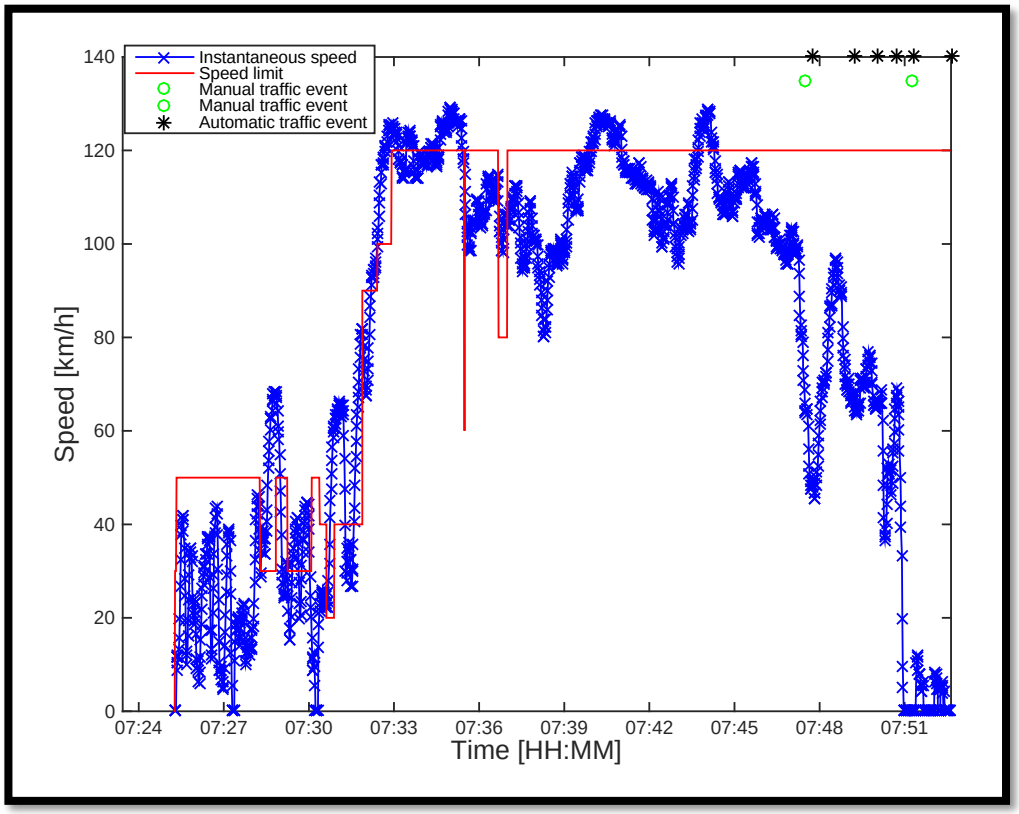

Illustration 10: Instantaneous speed, manual traffic events, automatic events, and Speed limit.

5. Assessments and results

In this chapter, we will test the developed solution. We will test the different operations, and we will analyse the results to determinate if it is a correct solution.

We have created a different test for each part of the application, but the most graphical one is a real time test. We have driven on a real road to prove the accuracy detecting congestion events. After taking some automatic and manual data, we have represented it on a map with the historic traffic of Google. As you can see on the next pictures, the results were very good.

In blue, we can see manual events, in red automatic events, and on the background Google traffic information.

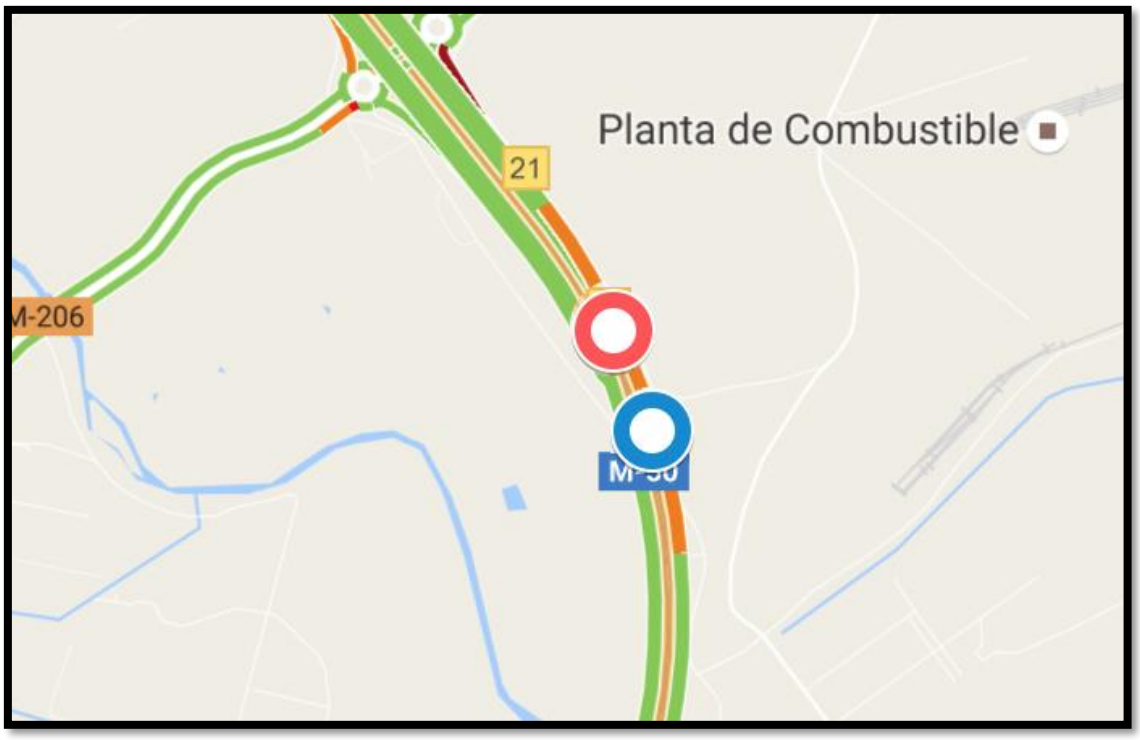

Illustration 11: Example of traffic detection. (Source Google Maps)

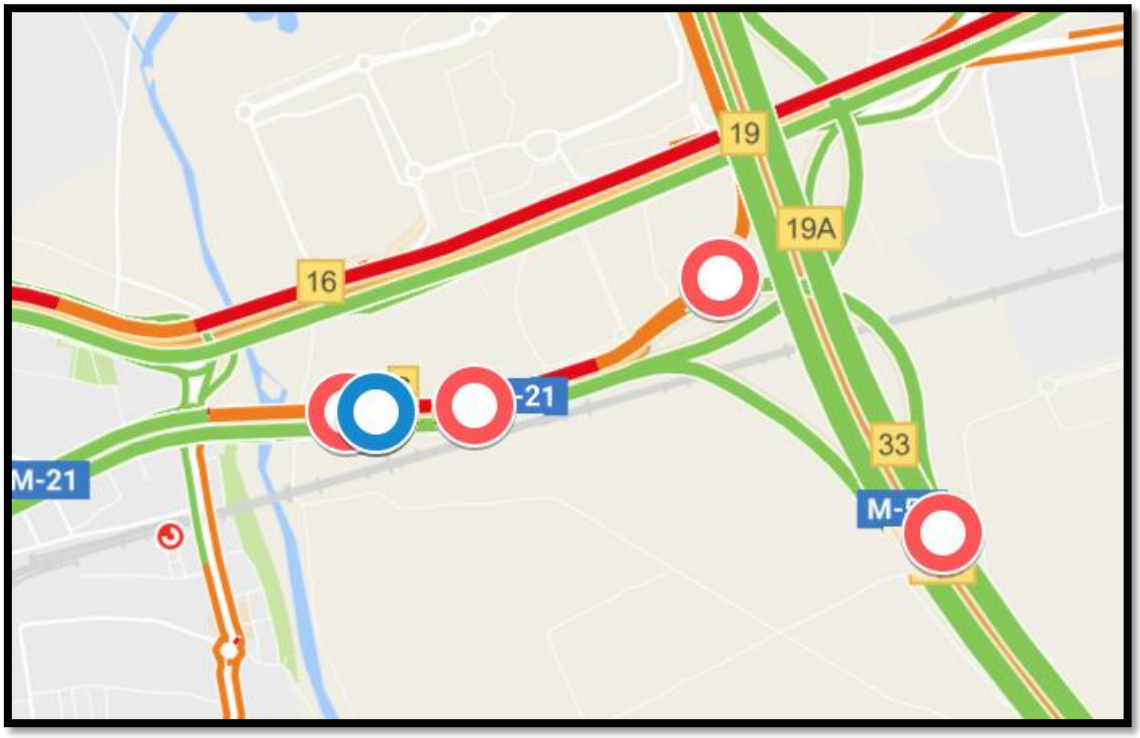

Illustration 12: **Another example of traffic detection. (Source Google Maps)**

6. Regulatory framework

In this chapter, we will have a look at the Europe and Spanish legislation that affects to this project.

Spanish Constitution of 1978: is the current supreme law of the Kingdom of Spain. On its 18th section we can read:

1) The right to honour, to personal and family privacy to the own image is guaranteed.
2) The home is inviolable. No entry or search may be made without the consent of the householder or a legal warrant, except in cases of flagrante delicto.
3) Secrecy of communications is guaranteed, particularly regarding postal, telegraphic an telephonic communications, except in the event of a court order.
4) The law shall restrict the use of data processing in order to guarantee the honour and personal and family privacy of citizens and the full exercise of their rights.

Based on this section, there are other laws such as intellectual property law, or users' privacy law that are important during the development of this project.

It is also very important to have a look at DGT laws. In Spain, it is forbidden to drive while you are using any type of mobile devices.

7. Conclusion

To conclude, we will try to give the conclusions about this project, for this, we will revise the main objectives and then we will give some tips about the future of this application.

The opportunity of making this project has been an academic, professional and personal challenge. The determination, during the development of the different parts, has taught me a new perspective to problem solving. This, for sure, will help me in the future.

It is important to say that the solution of this project is not the only one, but it is the one that fits with our resources and knowledge. As we have seen during this bachelor thesis, we compared different solution on different part of the project:

- We have compared different mobile operative systems. Finally, we chose Android.
- We have compared different sensors, and their capabilities, finally we chose accelerometer as the main one.
- We have compared different articles, and finally, we chose *TrafficSense.*
- We have compared different API's to take road or traffic data, and finally, we chose Here WeGo.

All the decisions have been based on different arguments during the development of this report.

7.1. Reach goals

During the start of this project, we establish some objectives. The achievement of then is a good way to measure the success.

7.1.1. Project objectives

Objective	Achievement
Main: Develop an Android application. It must solve traffic problem with an algorithm.	✓ We have developed an Android application to detect congestion events, on a successfully way.
Supporting: Concern about user's privacy.	✓ We have developed this project taking care of user's privacy. That is why we made a regulatory framework study.

Event detection will be local. The mobile device will detect events without sending information to the network.	✓ The algorithm has been developed to detect events thanks to device sensors as GPS or accelerometer. The information is sent only when an event has been detected.
The device will take care of the different resources.	✓ We have developed the application to save battery and data traffic.
Highlight the importance of the algorithm instead of the application.	✓ The main reason to develop this application is proving that this alternative way to detect traffic works.

Table 2: Project reaches goals.

7.1.2. Personal objectives

Objective	Achievement
Learn to develop mobile applications.	✓ We have developed an Android application successfully.
Study and learn agile development methodologies.	✓ We have developed the entire project using this type of methodology successfully.

Table 3: Personal reach goals.

7.2. The future of this application

There are so many possibilities to work and improve this application on a short term.

- Thanks to the hybrid data system, on the one hand, manual data from users, and on the other hand automatic data from the algorithm, we are able to establish different congestion levels (like Google maps).
- If we include our algorithm on a navigation application, where we can also use third party information, we will get more accurate traffic information, and extra services.
- We can optimise the calibration algorithm to improve the quality of the system.
- If we record traffic events on a server, we will be able to provide this information in real time to thousands of users. This could help all of them to avoid congestion situations.

Bibliografía[1]

[1] Definición: “Tránsito vehicular”, *Wikipedia*. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Tránsito_vehicular] (Consultado: 29 agosto 2016).

[2] Historia tráfico en España, *DGT*. [En línea]. Disponible en: [http://www.dgt.es/es/la-dgt/quienes-somos/historia/] (Consultado: 29 agosto 2016).

[3] Definición: “Congestión vehicular”, *Wikipedia*. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Congestión_vehicular] (Consultado 29 agosto 2016).

[4] “Plan de movilidad Urbana Sostenible de la ciudad de Madrid”, Ayuntamiento de Madrid, España. Diciembre 2014. [En línea]. Disponible en: [http://www.madrid.es/UnidadesDescentralizadas/UDCMovilidadTransportes/MOVILIDAD/PMUS_Madrid_2/PMUS%20Madrid/Plan%20de%20Movilidad%20de%20Madrid%20aprobacion%20final.pdf]

[5] Definición: “Google Traffic”, *Wikipedia*. [En línea]. Disponible en: [https://en.wikipedia.org/wiki/Google_Traffic] (Consultado: 30 agosto 2016).

[6] “Información sobre mapas”, *Google*. [En línea]. Disponible en: [https://support.google.com/maps/answer/3092439?hl=en&rd=2] (Consultado: 31 agosto 2016).

[7] D. Barth, Barth, “The bright side of sitting in traffic: Crowdsourcing road congestion data”, *Google blog*, 25-09-2009 [En línea]. Disponible en: [https://googleblog.blogspot.com.es/2009/08/bright-side-of-sitting-in-traffic.html]

[8] “Información sobre navegador”, *Waze*. [En línea]. Disponible en: [https://www.waze.com/es/about] (Consultado: 31 agosto 2016)

[9] “Estudio sobre el uso de dispositivos móviles”, *We are social*. [En línea]. Disponible en: [http://wearesocial.com/sg/] (Consultado: 31 agosto 2016).

[10] Definición: “GPS”, *Wikipedia*. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global] (Consultado: 1 Septiembre 2016).

[11] M. Arenas Mas, “Diseño e implementación de un sistema de adquisición de aceleraciones con procesamiento mediante microprocesador”, Proyecto fin de carrera, Dpto. de ingeniería electrónica, Universidad de Sevilla, Sevilla, España, 2008. [En línea]. Disponible en: [http://bibing.us.es/proyectos/abreproy/11638/fichero/Capitulo+4.pdf]

[1] Para la realización de esta bibliografía, se ha seguido el formato IEEE v8, conforme a lo indicado en: http://uc3m.libguides.com/guias_tematicas/citas_bibliograficas/IEEE#libro

[12] C.J. Bernardos Cano, Material asignatura Redes Inalámbricas y móviles. Universidad Carlos III de Madrid, Leganés, España, Curso 2014/2015.

[13] Informes telecomunicaciones, Comisión Nacional de los Mercados y la Competencia, España. [En línea]. Disponible en: [https://www.cnmc.es/]

[14] O. Cinar, *Android Apps with Eclipse,* 1st ed. Apress, 2012.

[15] S.F. Morrissey, *iOS Forensic Analisis for iPhone, iPad and iPod touch*, 1st ed. Apress, 2010.

[16] "Estudio sobre ventas en el mercado móvil" , *Kantar World Planet*. [En línea]. Disponible en: [http://www.kantarworldpanel.com/global/smartphone-os-market-share/] (Consultado: 1 de Septiembre 2016).

[17] "Web para desarrolladores de Android", *Google*. [En línea]. Disponible en: [https://developer.android.com/] (Consultado: 1 Septiembre 2016).

[18] "API de mapas", Web de servicios Google para desarrolladores", *Google*. [En línea]. Disponible en: [https://developers.google.com/maps/?hl=es] (Consultado. 1 septiembre 2016).

[19] "API de tráfico", Web de servicios HereWeGo para desarrolladores, *HereWeGo*. [En línea]. Disponible en: [https://developer.here.com] (Consultada: 1 de septiembre 2016).

[20] "Historia de Matlab", *Matlab*. [En línea]. Disponible en: [http://es.mathworks.com/products/matlab/?requestedDomain=es.mathworks.com] (Consultado: 1 de septiembre 2016).

[21] Constitución Española, Artículo 18. (BOE Nº 311, 29 diciembre 1978).

[22] Real decreto legislativo 1/1996, Ley de Propiedad Intelectual. (BOE Nº 97, 22 abril 1996).

[23] Ley 7/2010, General de la Comunicación Audiovisual. (BOE Nº 79, 1 enero 2010).

[24] Ley Orgánica 15/1999, de Protección de Datos de Carácter Personal. (BOE 298 , 14 diciembre 1999).

[25] Real decreto 1428/2003, Reglamento General de Circulación. (BOE Nº 306, 23 diciembre 2003).

[26] "Ilustración proceso Scrum", *Wikipedia*. [En línea]. Disponible en: [https://commons.wikimedia.org/wiki/File:Scrum_process.svg] (Consultado 2 septiembre 2016).

[27] P. Mohan, V.N. Padmanabhan and R. Ramjee. "TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones", *Technical Report MSR-TR-2008-59*, Bangalore, India, 2008. [En línea]. Disponible en: [https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2008-59.pdf]

[28] A. González Caballero. “Mapa Tráfico para TFG”. *Google Maps.* [En línea]. Disponible en: [https://www.google.com/maps/d/edit?mid=15o8styGdU3uPuD9uhjh_Ygrmgew]

[29] Definición: “Scrum”, *Scrum Alliance*. [En línea]. Disponible en: [https://www.scrumalliance.org] (Consultado: 16 septiembre 2016).

[30] R. Bhoraskar, “Traffic and Road Condition Estimation using Smartphone Sensors.”, Bachelor Thesis, Department of Computer Science and Engineering, Indian Institue of Technology, Bombay, Mumbai, India, 2011.

[31] S. Ayub, A. Bahraminisaab y B. Honary. “A sensor Fusion Method for Smartphone Orientation Stimation”, presentada en 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, Liverpool, Junio 2012. [En línea]. Disponible en: [http://www.cms.livjm.ac.uk/pgnet2012/Proceedings/Papers/1569603133.pdf]

[32] C. Campo Vázquez y C. García Rubio, Curso de Android de la asignatura Aplicaciones móviles, Universidad Carlos III de Madrid, Leganés, 2015.

[33] S. Gómez Oliver, *Curso Programación Android*, 3th ed. Publicación libre, 2015. [En línea]. Disponible en: [http://www.sgoliver.net/blog/curso-de-programacion-android/curso-en-pdf/]

[34] M. Bazeley, “Google acquires traffic info start-up Zipdash”, *SiliconBeat*. 30-03-2005. [En línea]. Disponible en: [http://www.siliconbeat.com/entries/2005/03/30/google_acquires_traffic_info_startup_zipdash.html] (Consultado 16 septiembre 2016).

[35] “APP Android DGT”, *Google Play Store*. [En línea]. Disponible en: [https://play.google.com/store/apps/details?id=es.quadram.dgt&hl=es] (Consultado 19 Septiembre 2016).

[36] R. Stoichkov. “Android Smartphone Application for Driving Style Recognition”, Project Thesis, Dpto. de ingeniería eléctrica y tecnologías de la información, Institute for Media Technology, Munich, Alemania, 12-07-2013. [En línea]. Disponible en: [http://www.eislab.fim.uni-passau.de/files/publications/students/Stoichkov-Projektarbeit.pdf] (Consultado 19 Septiembre 2016).

[37] N. Chintalacheruvu, V. Muthukumar, “Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems”, *Journal of transportation technologies*, Nº 2, pp. 305-314, 2012. [En línea]. Disponible en: [http://file.scirp.org/pdf/JTTs20120400004_70073830.pdf].

[38] F. Rose. “Pocket Monster: How DoCoMo’s wireless Internet service went from fad to phenom – and turned Japan into the first post-PC nation”, *Wired*, 01-09-01. [En línea]. Disponible en: [https://www.wired.com/2001/09/docomo/] (Consultado 19 Septiembre 2016).

[39] Definición: “Barómetro”, *Wikipedia*. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Barómetro] (Consultado: 19 Septiembre 2016).

[40] Definición: “Magnetómetro”, *Wikipedia*. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Magnetómetro] (Consultado: 19 Septiembre 2016).

[41] K. Schwaber, *Agile software development with scrum*, 1st ed. Pearson, 2001.

[42] Definición: “Casos de Uso”, *Wikipedia*. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Caso_de_uso] (Consultado: 19 Septiembre 2016).

[43] E. Gómez Martínez, “Aplicación Android para obtener información de las carreteras”, Trabajo de Fin de Grado, Departamento Ingeniería Telemática, Universidad Carlos III, Leganés, España, 2016.

[44] S. Suril Vijaykuma, S. Kumar Saha, J. Kumar Dutt, “Capítulo 3: Euler-Angle-Joints”, en *Dynamics of Tree-Type Robotic Systems*, 2013 ed. Springer, 2013.